Table of Contents

1. Introducing the topic – motivation, relevance

Introduction

Humor is one of the most sophisticated forms of human intelligence and communication. It has a potential to cause laughter, make people feel comfortable along with creating a cozier environment. In one form or in another humor is most often based on some form of incongruity. It is known to affect a persons' well-being, decision-making, and even psychological systems. While in some occasions humor becomes a defense mechanism to survive psychological stress, in others it becomes that much needed prerogative to etch a memory for a lifetime. For the four of us doing this project, we believe humor is an expression of human wisdom and originality, and shall remain until the end of humanity.

Expressing humor can be done in a zillion ways - writing, talking, sketching, conversing, and sometimes not doing anything. From the days where crowds of spectators would gather at a place to watch specially constructed gladiatorial shows or written on paper and published by newspapers, humor has evolved significantly. In a written context, it has evolved from publishing content to generating memes for social media; while in a spoken context, jokes were told on radio, at family functions, at events and now are being told on stage for live audiences as well. This recent trend in short is called 'Stand-up Comedy', which rose in popularity in the last decade due to the sheer exposure the comedians got through television shows, videos, and social media. At the end of the day, laughter is a universal language.

Motivation

The motivation for our team to explore humor detection in the field of Natural Language Processing stems from the ongoing question of what abilities and qualities make humans distinct from machines in today's technologically equipped world. As computers become more intelligent, there is a growing curiosity about the limitations of machines and the aspects of human behavior that they cannot replicate. Many often cite creativity, compassion, and humor as uniquely human traits that are challenging for machines to emulate.

Among these qualities, humor generation remains an intriguing and relatively unexplored problem. While humans possess the ability to produce laughter and use humor to evoke amusement in others, this capability appears to be exclusive to our species apart from a few primates. Various studies have shown that while primates, for example, can produce laughter, it is humans who engage in the deliberate use of humor for eliciting laughter from others. Despite these challenges, there has been some progress in the development of computational humor systems. These systems use a variety of techniques to detect and generate humor, including natural language processing, machine learning, and artificial intelligence.

Nonetheless, in this project, we are trying to explore several computational techniques to detect humor and present a unique use case through the insights generated.

Relevance

The practical applications of computational humor are vast and diverse. In recent years, platforms, products, and individuals have thrived by creating and delivering humorous yet engaging content. From e-commerce to computer-mediated communication and educational systems, humor has the potential to transform user experiences, enhance engagement, and contribute to the success of various domains in the modern world. Exploring and understanding computational techniques for humor detection can pave the way for innovative applications and improvements in these practical settings.

2. Major Use Cases & Applications

Automation of humor detection has interesting use cases in modern technologies, such as humanoid robots, chatbots, and virtual assistants. Furthermore, it is a powerful tool that can be used to influence people's attitudes and behaviors. For example, humor has been used to promote social change, sell products, and even win elections. Some major use cases are elaborated below:

Business world applications

Humor can be used to promote products and services, build relationships with customers, and create a more positive and engaging work environment. For example, many companies use humor for product promotion, capturing selective attention, and aiding in memorization. In an era where electronic commerce is booming, incorporating computer generated humor into presentations and advertisements can greatly enhance customer engagement and contribute to the success of marketing campaigns. Humor can also be used in customer service interactions to help resolve critical problems and build goodwill with customers.

General computer-mediated communication and interaction

Humor can be used to make computer-mediated communication (CMC) more engaging and enjoyable. For example, many chat rooms and online forums use humor to help users connect with each other and build relationships. Humor is being used to make human-computer interaction (HCI) more natural and intuitive. For example, many voice-activated assistants use humor to make interactions with users more fun and engaging.

Educational and edutainment systems

Incorporating humor in educational and edutainment systems is improving user experience, fostering engagement, and creating a more enjoyable and interactive environment, ultimately leading to enhanced learning outcomes, increased motivation in a positive atmosphere.

Automatic information presentation

In the age of the internet and personalized content, automatic information presentation is utilizing humor tailored to individual preferences to enhance the effectiveness of information delivery and promote product awareness. For example, many news websites and blogs use humor in their headlines and content links to help users understand and remember important news stories.

3. Background

Theoretical Background

Humor has been studied for centuries, and there are many different theories about what makes something funny. Some of the most common theories include:

Incongruity theory of Humor

This theory suggests that humor arises from a mismatch between what is expected and what actually happens. For example, a joke might be funny because it contains a surprise ending.

Relevance theory of Humor

This theory suggests that humor is funny when it is relevant to the listener's experiences and expectations. For example, a joke might be funny because it plays on a common stereotype or cultural reference.

Benign violation theory of Humor

This theory suggests that humor arises from a feeling of safety in the face of something that is mildly threatening or surprising. For example, a joke might be funny because it contains a mild insult or a gentle poke fun at a social norm.

By considering the above theoretical perspectives, researchers in the field of Natural Language Processing aimed to develop computational models and techniques that can detect and understand humor in textual data.

Technical Background

In the field of humor classification, several approaches have been developed using advanced machine learning techniques and Natural Language Processing (NLP) methods. These techniques aim to accurately classify text into humorous and non-humorous categories, allowing for a deeper understanding of humor in textual data.

One prominent approach is the utilization of deep learning models, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs have shown success in capturing local and global features in text by applying convolutional operations over input sequences. They can effectively learn hierarchical representations, making them suitable for humor classification tasks. RNNs, on the other hand, are capable of capturing sequential dependencies in text through their recurrent connections. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, which are variations of RNNs, have been widely used to model humor-related patterns and contextual information.

Another significant advancement is the integration of pre-trained language models, such as Transformer-based architectures like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). These models, trained on large amounts of text data, have demonstrated exceptional performance in various NLP tasks, including humor classification. By fine-tuning these pre-trained models on humor-specific datasets, they can effectively capture humor-related patterns, context, and semantic information.

Furthermore, ensemble learning techniques have been employed to enhance classification performance. Ensemble methods combine multiple individual models to make predictions, leveraging the diversity and complementary strengths of different models. By aggregating the predictions of multiple models, ensemble approaches can achieve higher accuracy and robustness in humor classification tasks.

Moreover, researchers have explored the use of attention mechanisms to identify key elements in text that contribute to humor. Attention mechanisms allow models to assign different weights to different parts of the input sequence, focusing on the most relevant information for classification. By incorporating attention mechanisms into humor classification models, they can effectively identify humor-related cues and improve overall classification accuracy.

4. State of the Art (Literature Survey)

The field of humor detection in Natural Language Processing (NLP) has witnessed significant advancements in recent years. The field evolved from using traditional feature-based approaches to the application of deep learning techniques and large language models. The utilization of convolutional networks, recurrent networks, and Transformer-based architectures has significantly improved the accuracy and effectiveness of humor detection models. Some literature gathered is as below:

Early Research

Early research focused on utilizing N-grams, stylistic features, and Random Forest classifiers with Word2Vec embeddings and Human Centric Features to identify humor in text. These initial approaches, such as those proposed by Taylor and Mazlack (2004), Mihalcea and Strapparava (2006), and Yang et al. (2015), paved the way for further exploration.

Use of Deep Learning Techniques

In more recent studies, deep learning techniques have been employed for humor detection. Chen and Soo (2018) combined Convolutional Neural Networks with Highway Networks, while de Oliveira and Rodrigo (2015) utilized Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks to analyze humorous text.

Use of Transformer Architecture

The emergence of the Transformer architecture, as introduced by Vaswani et al. (2017), has sparked a shift in research towards fine-tuning large language models for humor classification. Researchers, such as Blinov et al. (2019), Weller and Seppi (2019), and Annamoradnejad (2020), have leveraged these powerful language models to improve humor detection performance on various datasets.

Use of Large Language Models

Furthermore, Peyrard et al. (2021) have recently explored the use of large language models to distinguish between serious and humorous sentences. Their approach involves analyzing Transformer attention heads to uncover evidence of humor sensitivity.

Use of Recurrent Neural Networks

In parallel efforts, Pitsilis et al. (2018) demonstrated the capability of Recurrent Neural Networks (RNNs) to detect offensive language in Twitter data. Building upon this work, Barbieri et al. (2020) enhanced the results by incorporating Transformer-based architectures, which allowed for improved offensive language detection.

Researchers continue to explore innovative methods and fine-tuning strategies to further enhance the performance of humor classification systems. In this paper, we are trying to utilize the application of humor detection in the entertainment sector, which includes movies, TV shows, streamed videos, etc.

5. Limitations & Challenges of the State of the Art

Limitations

The state-of-the-art approaches for humor detection are based on machine learning techniques. These methods are trained on large datasets of labeled text, such as jokes that have been rated by human judges for humor. The models are then able to predict whether a new piece of text is humorous or not.

Limitation due to training data

However, these methods have a number of limitations. First, they are only as good as the training data they are trained on. If the training data does not contain a particular type of humor, the model will not be able to detect that type of humor. Second, these methods are not always accurate. Even the best models make mistakes, and they may sometimes incorrectly classify a piece of text as humorous or not humorous.

Limitations due to the Nature of Humor

There are a number of reasons why humor detection is such a difficult task. Firstly, humor is subjective. What one person finds funny, another person may not. Secondly, humor is often based on cultural references or shared experiences, which can make it difficult to detect humor in text that is not familiar to the reader. Finally, humor can be subtle and difficult to identify. Sometimes, the humor is in the way something is said, rather than in the words themselves.

Challenges

In addition to the limitations mentioned above, there are a number of other challenges that need to be addressed in order to develop more accurate and effective humor detection systems. These challenges include:

The need for better datasets

The current state-of-the-art methods for humor detection are trained on datasets of labeled text, such as jokes that have been rated by human judges for humor. However, these datasets are often small and biased, which can limit the accuracy of the models.

The need for better understanding of humor

The current state-of-the-art methods for humor detection are based on machine learning techniques, but these techniques do not fully understand how humor works. In order to develop more accurate and effective humor detection systems, we need to better understand the cognitive and linguistic processes that underlie humor.

The need for better methods for incorporating context

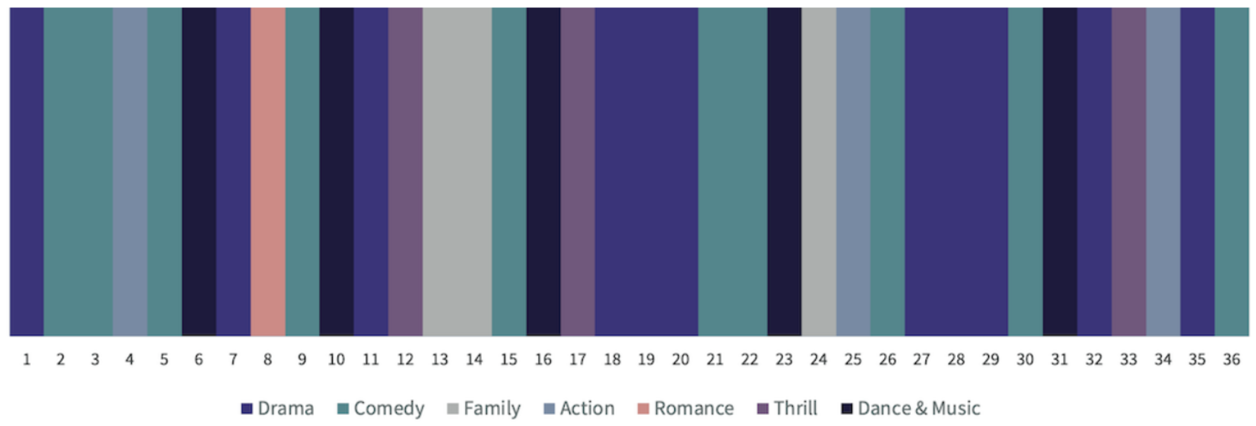
Humor is often influenced by context, such as the speaker's tone of voice, the audience's cultural background, and the current situation. In order to develop more accurate and effective humor detection systems, we need to develop better methods for incorporating context into the analysis.

Despite the limitations and challenges, machine learning methods for humor detection have made significant progress in recent years. The accuracy of these methods has improved, and they are now being used in a variety of applications, such as social media filtering, generating educational materials and personalized recommendations. As research in this area continues, we can expect to see even more accurate and effective humor detection systems being developed.

6. Application - Entertainment Sector

Can we make a computer trained to detect various human emotions, watch a film just like a human and draw insights from it? With this question in our mind, we started off to research the application of humor detection in the entertainment industry.

In this context, the use case we came up with is to detect various emotions such as Drama, Comedy, Family, Action, Romance, Thrill, Dance and Music in a full length movie and draw insights on the length (run-time) of each of the mentioned emotions. The output could look something like the following:



In the above picture, the X-axis represents a 5 minute run-time of a movie. The total movie length is 180 minutes, which is typical to most Indian movies. The first 5 minutes is characterized as Drama, the next 10 minutes is comedy, the next 5 minutes is Family and so-on.

A simple graph like the above can help multiple technicians in the entertainment industry in a zillion ways. First, this gives an overall view of the script to the writer. Second, a director could analyze the above graph and make appropriate changes in the run-time of a particular emotion. For example, if a particular emotion is dragged for a longer time, the audience could get bored. Our idea could help a director ask for appropriate script changes. Third, a producer could decide on the required cast and crew to shoot the movie and arrive at an appropriate schedule. Finally, after the movie completes shooting, a distributor can decide on the price that can be paid for buying such a movie.

While the above is the larger scope of our idea, in this paper, we aim to detect the emotion - comedy based on the script. To accomplish the same, we trained our model with humorous and non-humorous sentences. Then, we gave the script of FRIENDS, the most popular humorous television show of all time and tried to detect the humor in the script. The results were encouraging and the analysis is covered in the further sections.

7. Our Proposed Solution

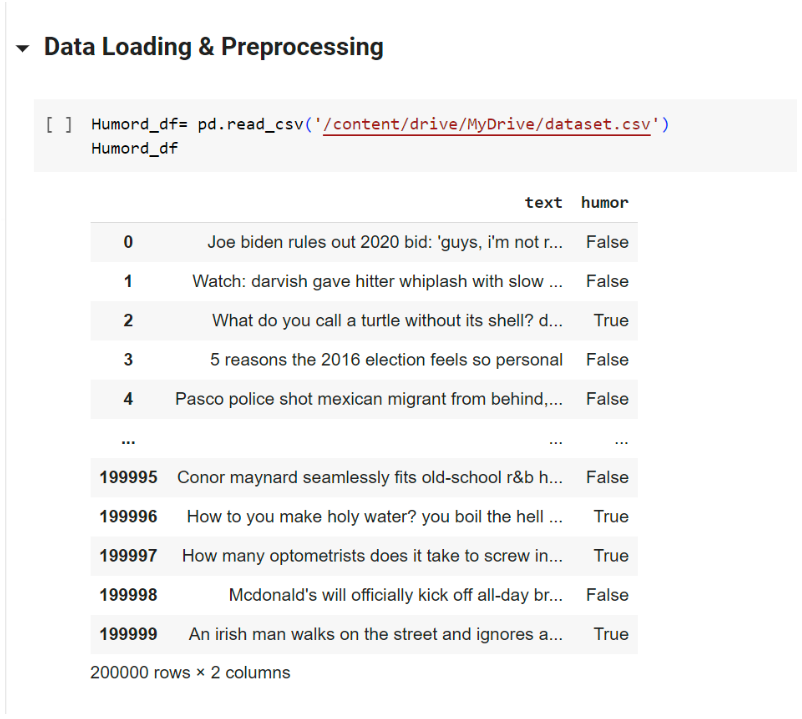
Our Dataset

For our train and test data we have used the existing data set provided at Humor Detection | Kaggle.

This pre processed data set is having 200K observations with two features “text” and “humor”

“text” ⇒ This feature is having sentence or dialogue
“humour” ⇒ This feature is having integer values
“0” signifies “Non-Humor” and “1” signifies “Humor”

Top and Bottom 5 observations with text and humor feature values.

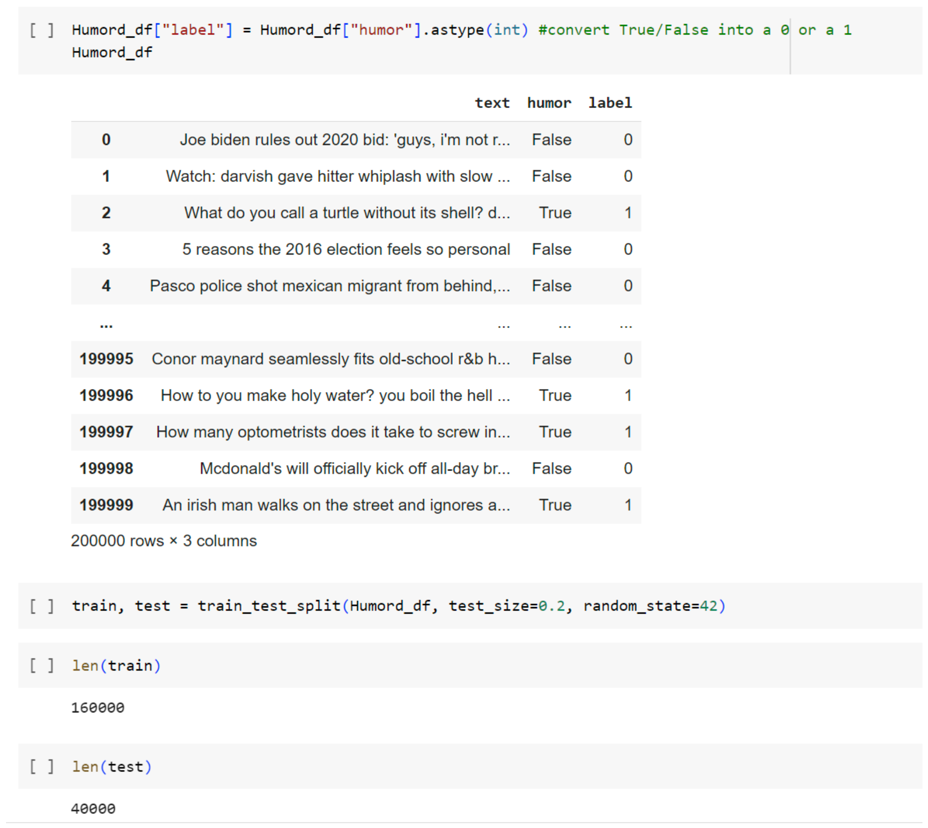

Data Loading & Preprocessing

```
[ ] Humord_df= pd.read_csv('/content/drive/MyDrive/dataset.csv')
    Humord_df
```

	text	humor
0	Joe biden rules out 2020 bid: 'guys, i'm not r...	False
1	Watch: darvish gave hitter whiplash with slow ...	False
2	What do you call a turtle without its shell? d...	True
3	5 reasons the 2016 election feels so personal	False
4	Pasco police shot mexican migrant from behind,...	False
...	...	...
199995	Conor maynard seamlessly fits old-school r&b h...	False
199996	How to you make holy water? you boil the hell ...	True
199997	How many optometrists does it take to screw in...	True
199998	Mcdonald's will officially kick off all-day br...	False
199999	An irish man walks on the street and ignores a...	True

200000 rows × 2 columns

Append another feature called “label” by converting humor True/Value conversion to 0 or 1



```
[ ] Humord_df["label"] = Humord_df["humor"].astype(int) #convert True/False into a 0 or a 1
    Humord_df
```

```
                                                     text  humor  label
0        Joe biden rules out 2020 bid: 'guys, i'm not r...  False      0
1        Watch: darvish gave hitter whiplash with slow ...  False      0
2          What do you call a turtle without its shell? d...   True      1
3             5 reasons the 2016 election feels so personal  False      0
4        Pasco police shot mexican migrant from behind,...  False      0
...                                                    ...    ...    ...
199995   Conor maynard seamlessly fits old-school r&b h...  False      0
199996     How to you make holy water? you boil the hell ...   True      1
199997   How many optometrists does it take to screw in...   True      1
199998        Mcdonald's will officially kick off all-day br...  False      0
199999     An irish man walks on the street and ignores a...   True      1
200000 rows × 3 columns
```

```
[ ] train, test = train_test_split(Humord_df, test_size=0.2, random_state=42)
```

```
[ ] len(train)

    160000
```

```
[ ] len(test)

    40000
```

Our Model

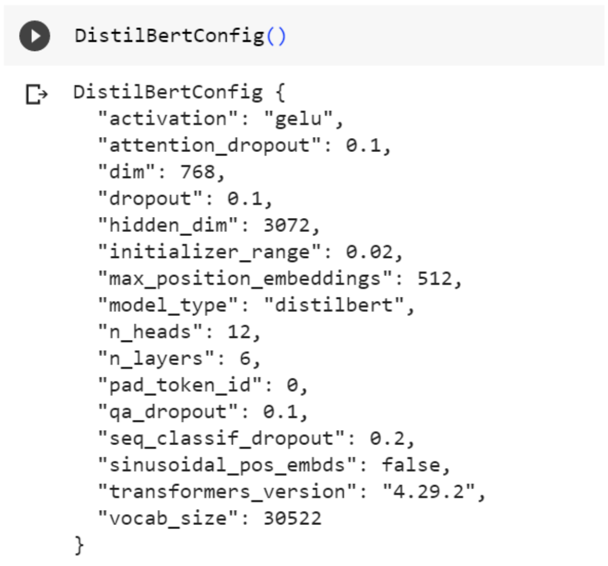
Our problem is to define if a given sentence is Humorous statement or not and this deals with the NLP (Text analytics) classification techniques.

We zeroed on to use distilbert-base-uncased which is a specific variant of the DistillBERT pre-trained model which was built on a large corpus of text data.

DistillBERTs a variant of the BERT (Bidirectional Encoder Representations from Transformers) model developed by Hugging Face.

DistilBERT achieves this by applying a technique called knowledge distillation, where a larger pre-trained model (such as BERT) is used to train a smaller model (DistilBERT).

Default Base Model Configuration parameters:



```
DistilBertConfig()
```

```
DistilBertConfig {
  "activation": "gelu",
  "attention_dropout": 0.1,
  "dim": 768,
  "dropout": 0.1,
  "hidden_dim": 3072,
  "initializer_range": 0.02,
  "max_position_embeddings": 512,
  "model_type": "distilbert",
  "n_heads": 12,
  "n_layers": 6,
  "pad_token_id": 0,
  "qa_dropout": 0.1,
  "seq_classif_dropout": 0.2,
  "sinusoidal_pos_embds": false,
  "transformers_version": "4.29.2",
  "vocab_size": 30522
}
```

- ***activation:*** The activation function used in the model. In this case, it's the GELU activation function.
- ***attention_dropout:*** The dropout probability defined as 0.1 for the attention mechanism in the model.
- ***dim***: The dimensionality of the model's hidden layers. In this case, it's set to 768.
- ***dropout:*** The dropout probability for the non-attention layers in the model.
- ***hidden_dim:*** The dimensionality of the intermediate feed-forward layer in the model.
- ***initializer_range:*** The range for initializing the model's parameters.
- ***max_position_embeddings:*** The maximum sequence length that the model can handle.
- ***model_type:*** The type of the model, which is "distilbert" in this case.
- ***n_heads:*** The number of attention heads in the model.
- ***n_layers:*** The number of transformer layers in the model.
- ***pad_token_id:*** The token ID used for padding sequences.
- ***qa_dropout:*** The dropout probability for the question-answering (QA) task.
- ***seq_classif_dropout:*** The dropout probability for the sequence classification task.
- ***sinusoidal_pos_embds:*** Whether to use sinusoidal positional embeddings in the model. In this case, it's set to false.

- ***transformers_version:*** The version of the transformers library used.
- vocab_size: The size of the vocabulary used by the model.

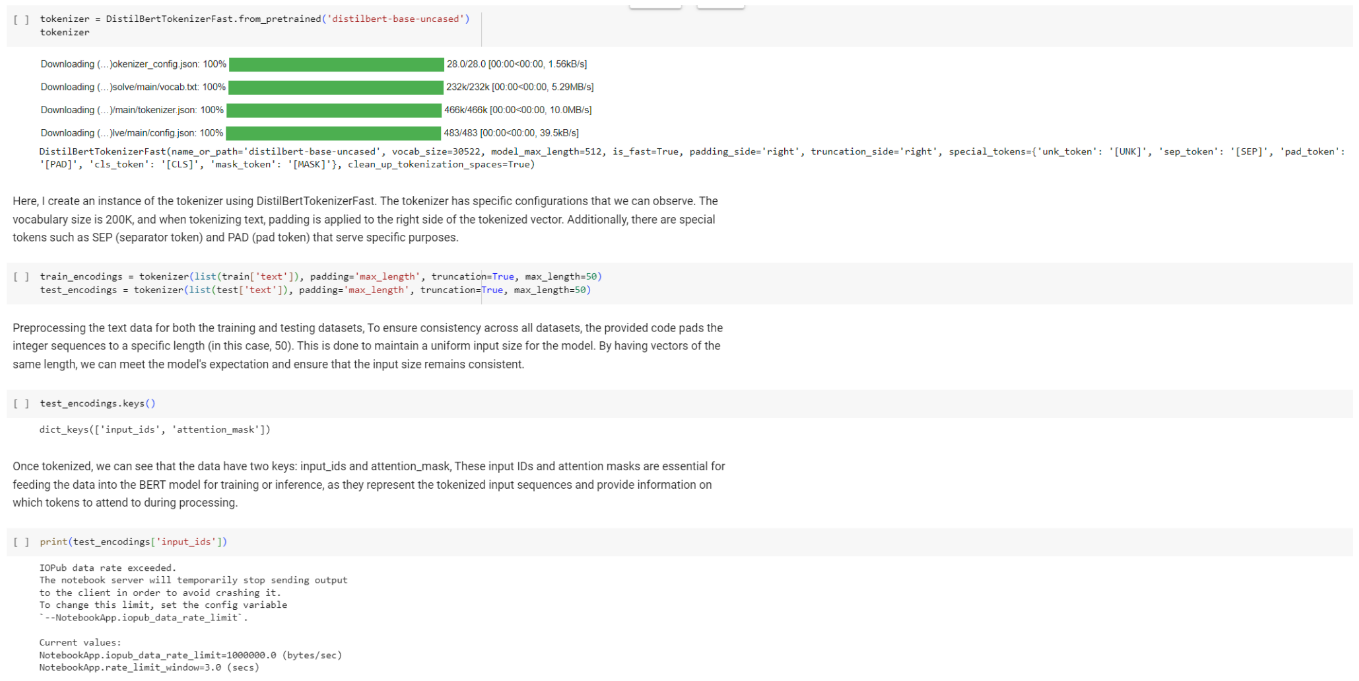
Model Implementation on train data set

After Data Loading and Pre-Processing steps

Step 1: Tokenizing the text to integer vectors

DistilBertTokenizerFast is a tokenizer class available as part of Hugging Face Transformers library and we have used this class as it simplifies the process of tokenizing text and preparing it for input to the DistilBERT model. It handles tokenization, special tokens, vocabulary, encoding, decoding, and other related tasks efficiently, making it a convenient tool for NLP tasks that involve the use of DistilBERT.

Please refer to the below code snippet:



```
[ ] tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
    tokenizer
```

```
Downloading (…)okenizer_config.json: 100% 28.0/28.0 [00:00<00:00, 1.56kB/s]
Downloading (…)solve/main/vocab.txt: 100% 232k/232k [00:00<00:00, 5.29MB/s]
Downloading (…)/main/tokenizer.json: 100% 466k/466k [00:00<00:00, 10.0MB/s]
Downloading (…)lve/main/config.json: 100% 483/483 [00:00<00:00, 39.5kB/s]
DistilBertTokenizerFast(name_or_path='distilbert-base-uncased', vocab_size=30522, model_max_length=512, is_fast=True, padding_side='right', truncation_side='right', special_tokens={'unk_token': '[UNK]', 'sep_token': '[SEP]', 'pad_token':
'[PAD]', 'cls_token': '[CLS]', 'mask_token': '[MASK]'}, clean_up_tokenization_spaces=True)
```

Here, I create an instance of the tokenizer using DistilBertTokenizerFast. The tokenizer has specific configurations that we can observe. The vocabulary size is 200K, and when tokenizing text, padding is applied to the right side of the tokenized vector. Additionally, there are special tokens such as SEP (separator token) and PAD (pad token) that serve specific purposes.

```
[ ] train_encodings = tokenizer(list(train['text']), padding='max_length', truncation=True, max_length=50)
    test_encodings = tokenizer(list(test['text']), padding='max_length', truncation=True, max_length=50)
```

Preprocessing the text data for both the training and testing datasets, To ensure consistency across all datasets, the provided code pads the integer sequences to a specific length (in this case, 50). This is done to maintain a uniform input size for the model. By having vectors of the same length, we can meet the model's expectation and ensure that the input size remains consistent.

```
[ ] test_encodings.keys()
```

```
dict_keys(['input_ids', 'attention_mask'])
```

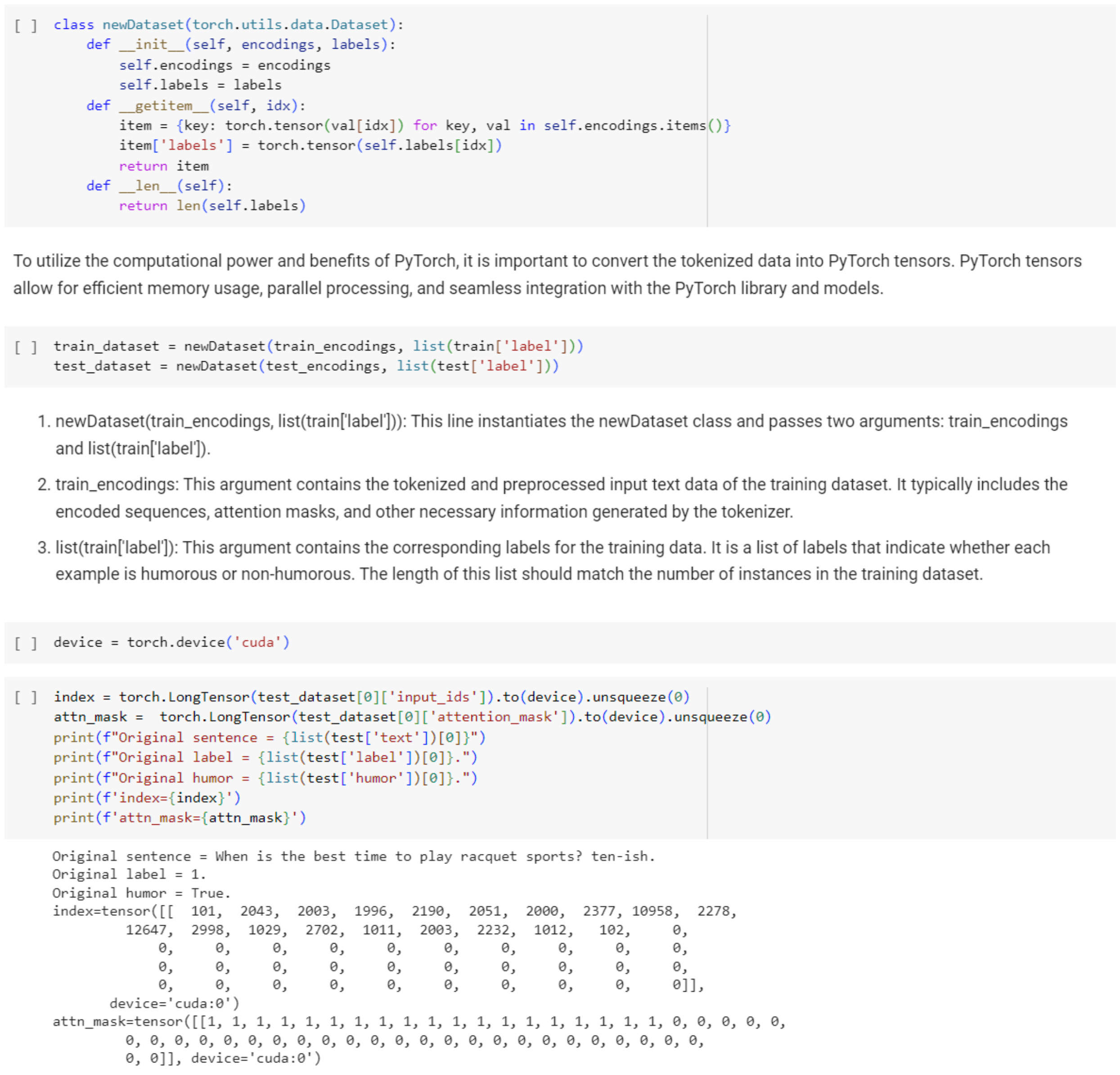
Once tokenized, we can see that the data have two keys: input_ids and attention_mask, These input IDs and attention masks are essential for feeding the data into the BERT model for training or inference, as they represent the tokenized input sequences and provide information on which tokens to attend to during processing.

```
[ ] print(test_encodings['input_ids'])
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Step 2: Model Training



```
[ ]  class newDataset(torch.utils.data.Dataset):
         def __init__(self, encodings, labels):
             self.encodings = encodings
             self.labels = labels
         def __getitem__(self, idx):
             item = {key: torch.tensor(val[idx]) for key, val in self.encodings.items()}
             item['labels'] = torch.tensor(self.labels[idx])
             return item
         def __len__(self):
             return len(self.labels)
```

To utilize the computational power and benefits of PyTorch, it is important to convert the tokenized data into PyTorch tensors. PyTorch tensors allow for efficient memory usage, parallel processing, and seamless integration with the PyTorch library and models.

```
[ ]  train_dataset = newDataset(train_encodings, list(train['label']))
     test_dataset = newDataset(test_encodings, list(test['label']))
```

1. newDataset(train_encodings, list(train['label'])): This line instantiates the newDataset class and passes two arguments: train_encodings and list(train['label']).
2. train_encodings: This argument contains the tokenized and preprocessed input text data of the training dataset. It typically includes the encoded sequences, attention masks, and other necessary information generated by the tokenizer.
3. list(train['label']): This argument contains the corresponding labels for the training data. It is a list of labels that indicate whether each example is humorous or non-humorous. The length of this list should match the number of instances in the training dataset.

```
[ ]  device = torch.device('cuda')
```

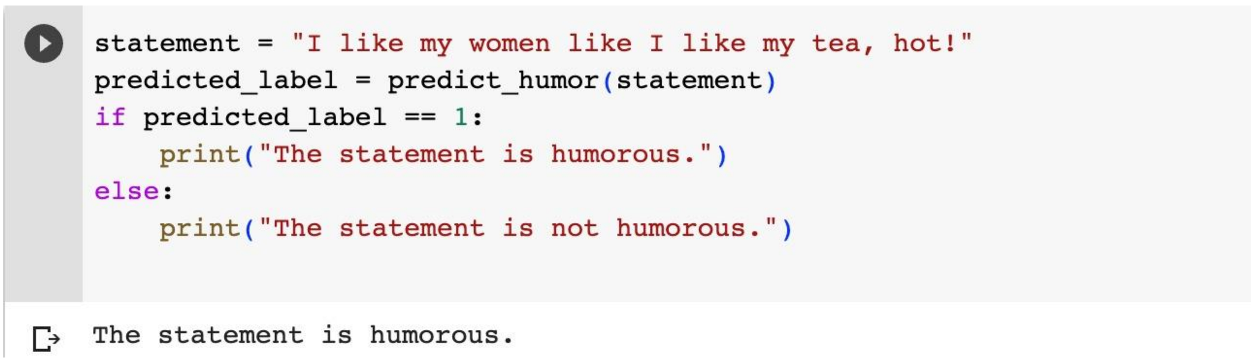
```
[ ]  index = torch.LongTensor(test_dataset[0]['input_ids']).to(device).unsqueeze(0)
     attn_mask =  torch.LongTensor(test_dataset[0]['attention_mask']).to(device).unsqueeze(0)
     print(f"Original sentence = {list(test['text'])[0]}")
     print(f"Original label = {list(test['label'])[0]}.")
     print(f"Original humor = {list(test['humor'])[0]}.")
     print(f'index={index}')
     print(f'attn_mask={attn_mask}')
```

```
Original sentence = When is the best time to play racquet sports? ten-ish.
Original label = 1.
Original humor = True.
index=tensor([[  101,  2043,  2003,  1996,  2190,  2051,  2000,  2377, 10958,  2278,
         12647,  2998,  1029,  2702,  1011,  2003,  2232,  1012,   102,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
             0,     0,     0,     0,     0,     0,     0,     0,     0,     0]],
       device='cuda:0')
attn_mask=tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0]], device='cuda:0')
```

Our Experiments

Experiment 1:

We have given input as a statement and predicted with our model. The following is the output which is correct as the given statement is a humorous one.

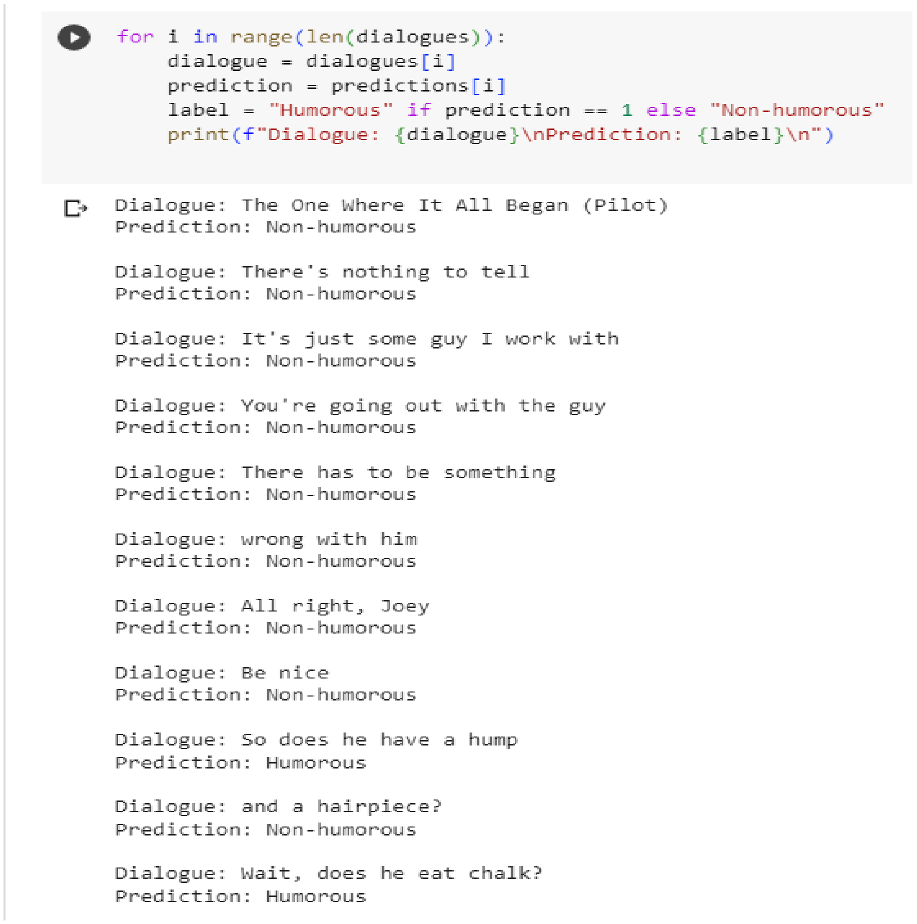


```python
statement = "I like my women like I like my tea, hot!"
predicted_label = predict_humor(statement)
if predicted_label == 1:
    print("The statement is humorous.")
else:
    print("The statement is not humorous.")
```

```
The statement is humorous.
```

Experiment 2:

We have given test data from one episode of the FRIENDS script and the predicted output along with code snippet is as follows:



```python
for i in range(len(dialogues)):
    dialogue = dialogues[i]
    prediction = predictions[i]
    label = "Humorous" if prediction == 1 else "Non-humorous"
    print(f"Dialogue: {dialogue}\nPrediction: {label}\n")
```

```
Dialogue: The One Where It All Began (Pilot)
Prediction: Non-humorous

Dialogue: There's nothing to tell
Prediction: Non-humorous

Dialogue: It's just some guy I work with
Prediction: Non-humorous

Dialogue: You're going out with the guy
Prediction: Non-humorous

Dialogue: There has to be something
Prediction: Non-humorous

Dialogue: wrong with him
Prediction: Non-humorous

Dialogue: All right, Joey
Prediction: Non-humorous

Dialogue: Be nice
Prediction: Non-humorous

Dialogue: So does he have a hump
Prediction: Humorous

Dialogue: and a hairpiece?
Prediction: Non-humorous

Dialogue: Wait, does he eat chalk?
Prediction: Humorous
```

Evaluation parameters

As Part of our assignment we are using the DistillBERT model as a sequence classification. The below are commonly available evaluation parameters available with the DestillBERT model.

Accuracy

Accuracy measures the proportion of correctly predicted labels out of the total number of samples. It is a widely used evaluation metric that provides an overall measure of the model's performance.

Precision

Precision is the ratio of true positive predictions to the total number of positive predictions made by the model. It indicates how many of the predicted positive samples are actually true positives.

Recall

Recall, also known as sensitivity or true positive rate, is the ratio of true positive predictions to the total number of actual positive samples. It measures the model's ability to correctly identify positive samples.

F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a balanced measure of both precision and recall and is commonly used when the classes are imbalanced.

Area Under the ROC Curve (AUC-ROC)

The AUC-ROC metric measures the trade-off between true positive rate (TPR) and false positive rate (FPR) at various classification thresholds. It provides an aggregate measure of the model's performance across different threshold settings.

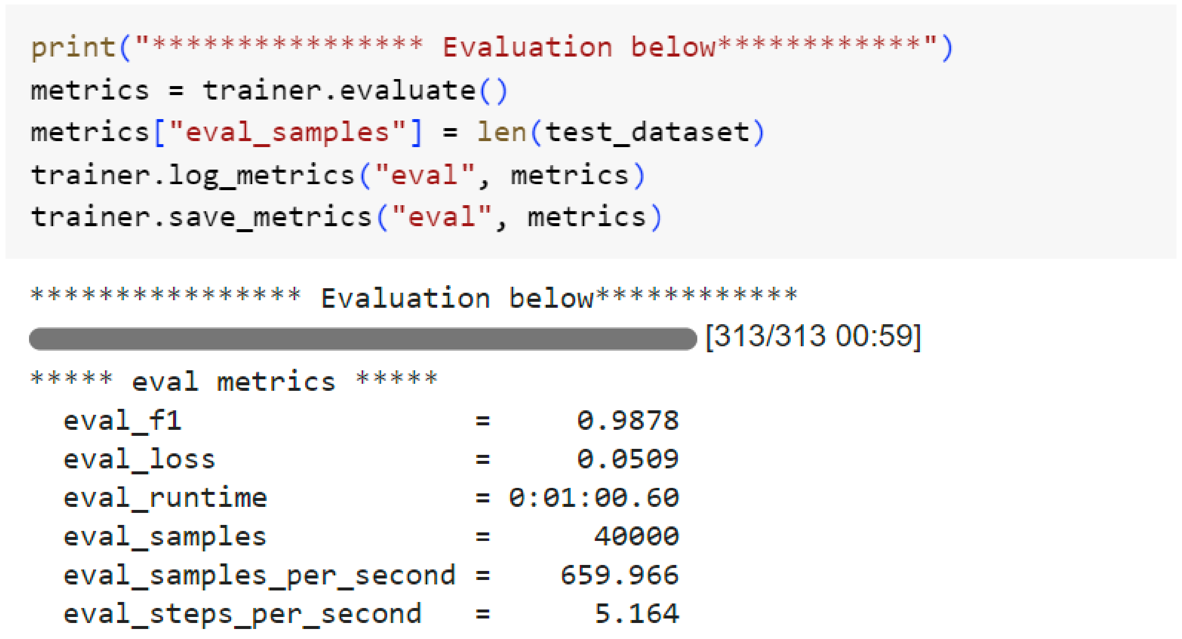
Confusion Matrix

A confusion matrix provides a detailed breakdown of the model's predictions for each class, showing the number of true positives, true negatives, false positives, and false negatives. It is useful for understanding the model's performance on individual classes and identifying specific areas for improvement.

As part of the assignment (Humor Detection) we are using F1 Score to determine as a means to objectively measure, compare, and monitor the performance of the Model we used and the below is the code snippet we have written

```
metric = load_metric("f1") #can also put "accuracy" if desired.
def compute_metrics(eval_pred):
logits, labels = eval_pred
predictions = np.argmax(logits, axis=-1)
return metric.compute(predictions=predictions, references=labels)
```

The below code showing the F1 score on the Train data set which is 98%

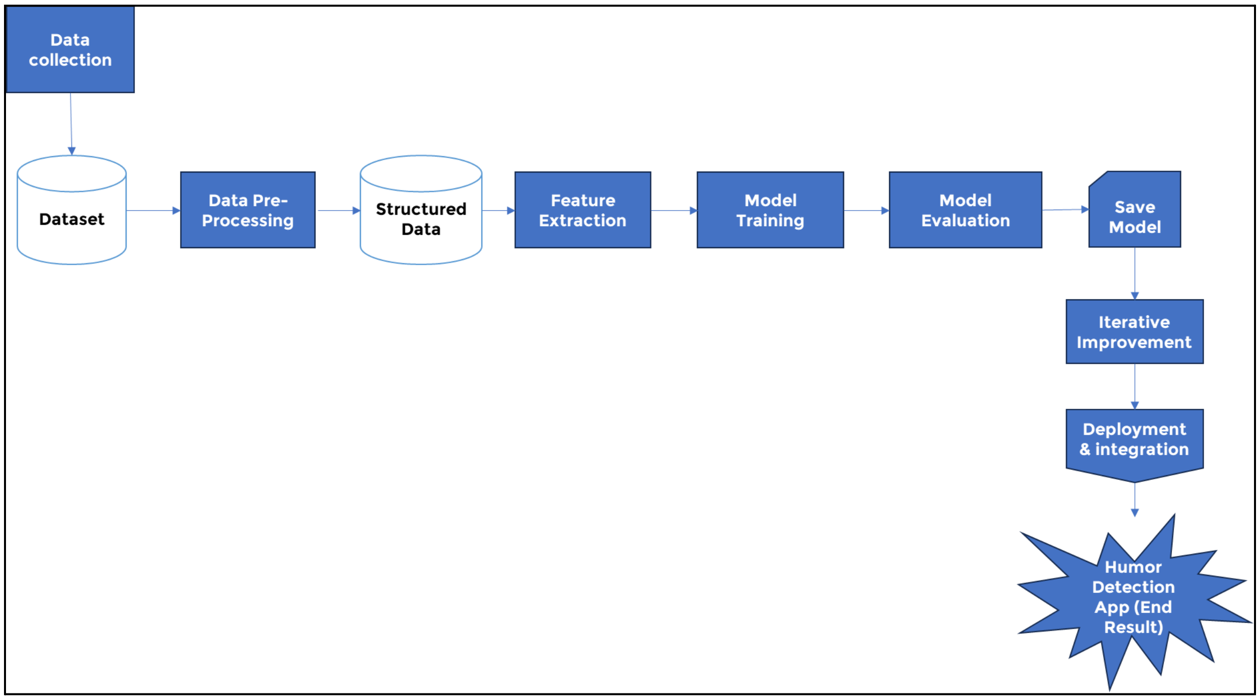


```
print("**************** Evaluation below************")
metrics = trainer.evaluate()
metrics["eval_samples"] = len(test_dataset)
trainer.log_metrics("eval", metrics)
trainer.save_metrics("eval", metrics)
```

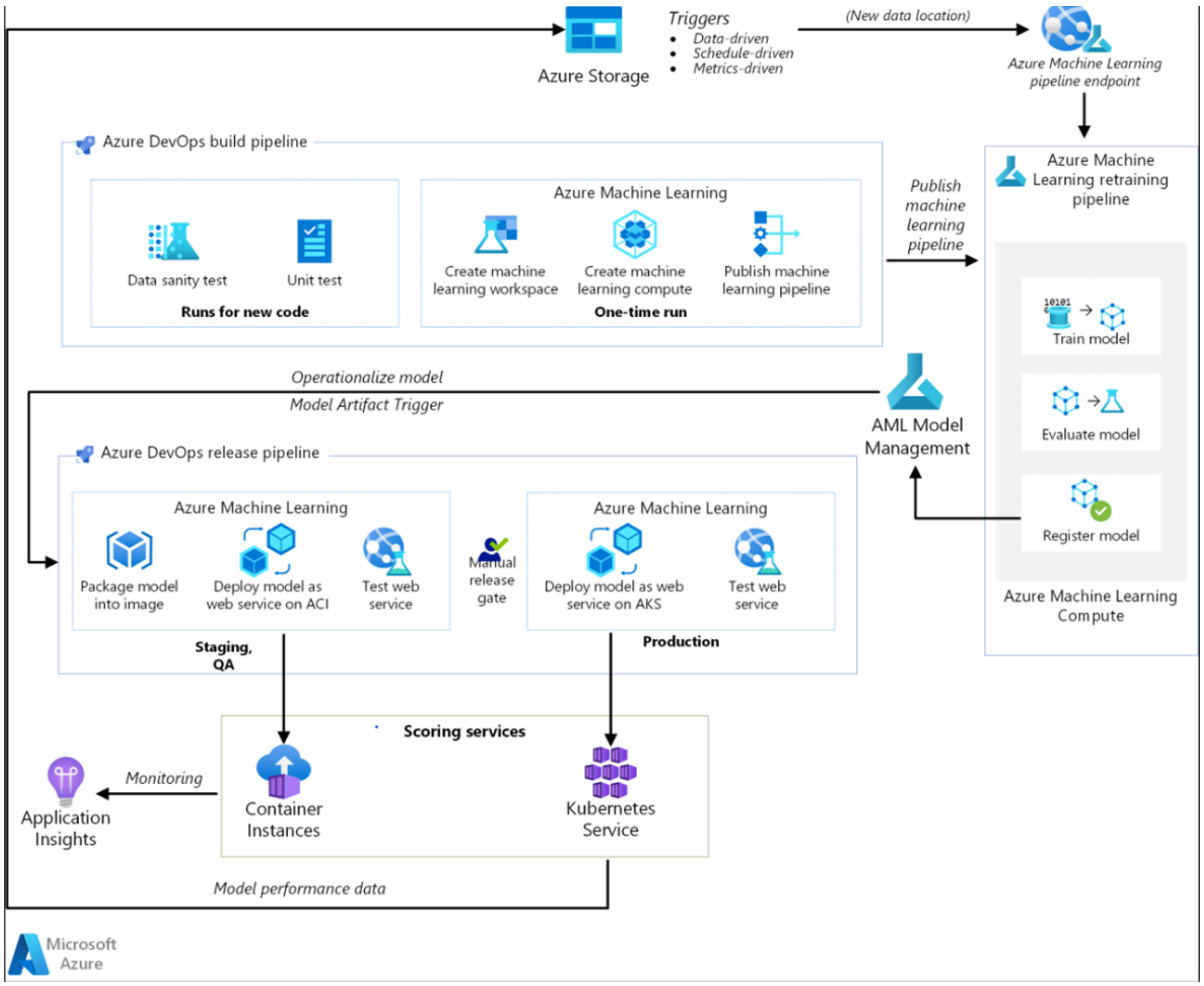
```
**************** Evaluation below************
[313/313 00:59]
***** eval metrics *****
  eval_f1                 =     0.9878
  eval_loss               =     0.0509
  eval_runtime            = 0:01:00.60
  eval_samples            =      40000
  eval_samples_per_second =    659.966
  eval_steps_per_second   =      5.164
```

8. Detailed Method, Pipeline & Architecture

The below architecture is showing a generic architecture which we used to build our Humour detection NLP model



The below Architecture showing the Azure ML Pipeline architecture



The detailed code walkthrough has been uploaded in the video as part of the assignment.

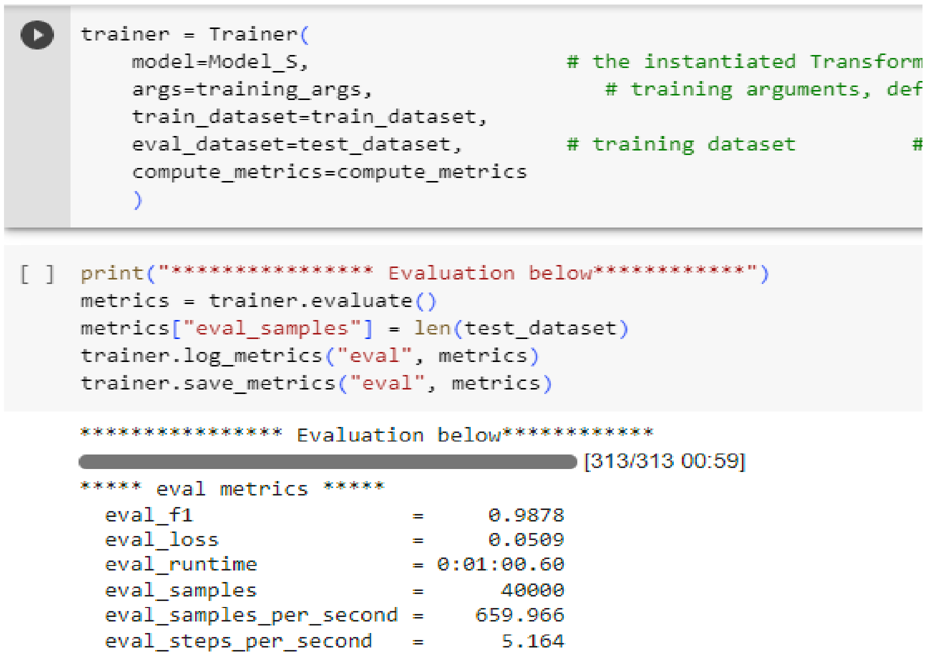
9. Key Findings and Conclusion

Performance of the Model:

As we have used one of the best suited models from the Hugging Face library. Nearly 1 hour has been taken to build the non-humorous and humorous training data and be able to predict the out on number of test data with just a click.

Test Model F1 Score is Amazing 98%:

We see that F1 score is 98% on test data based on model trained

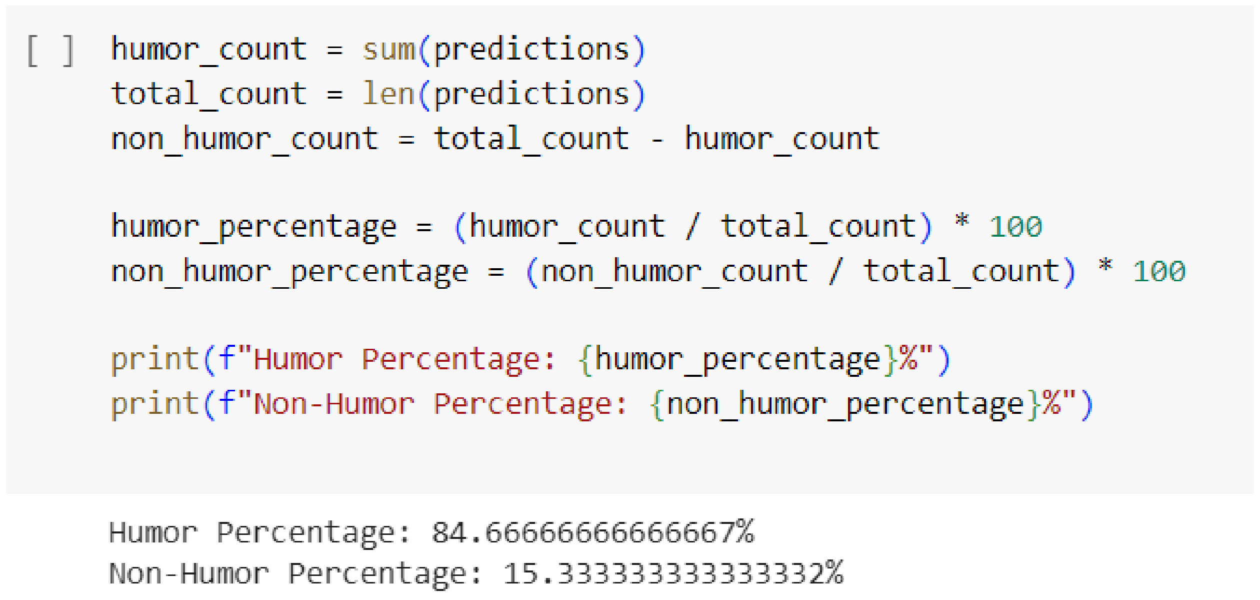


```
trainer = Trainer(
    model=Model_S,                # the instantiated Transform
    args=training_args,              # training arguments, def
    train_dataset=train_dataset,
    eval_dataset=test_dataset,    # training dataset         #
    compute_metrics=compute_metrics
    )
```

```
[ ] print("**************** Evaluation below************")
    metrics = trainer.evaluate()
    metrics["eval_samples"] = len(test_dataset)
    trainer.log_metrics("eval", metrics)
    trainer.save_metrics("eval", metrics)
```

```
**************** Evaluation below************
                                        [313/313 00:59]
***** eval metrics *****
  eval_f1                  =     0.9878
  eval_loss                =     0.0509
  eval_runtime             = 0:01:00.60
  eval_samples             =      40000
  eval_samples_per_second  =    659.966
  eval_steps_per_second    =      5.164
```

FRIENDS script prediction



```
[ ]  humor_count = sum(predictions)
     total_count = len(predictions)
     non_humor_count = total_count - humor_count

     humor_percentage = (humor_count / total_count) * 100
     non_humor_percentage = (non_humor_count / total_count) * 100

     print(f"Humor Percentage: {humor_percentage}%")
     print(f"Non-Humor Percentage: {non_humor_percentage}%")
```

```
Humor Percentage: 84.66666666666667%
Non-Humor Percentage: 15.333333333333332%
```

The above clearly shows that about 85% of the script of FRIENDS TV Show is humorous. This exactly fits in with the general human intuition also where majority of the people watching the series watch it for tickling their humor nerve.

Conclusion:

Based on three experiments

1. 98% accuracy with Test data
2. 100% accuracy with single statement
3. Our model is able to predict the humorous content on a full episode as well.

We can improve the accuracy of the model if we can train with more diverse and huge data sets so that accuracy can be improved.

Finally, we think we are on the right track to be able to give a report like the one mentioned in the applications section (Section 6). Humor part of it is solved.

10. Further Reading

Humor Detection ability for a model becomes quintessential while the research continues to build a robot with human-like abilities and capabilities. There is a lot of research already done and a lot more in the process. The following are the resources that can be referred for deeper understanding of our idea.

Research Papers along with Datasets

1. Issa Annamoradnejad, March 9, 2021, "ColBERT dataset - 200k short texts for humor detection", IEEE Dataport, doi: https://dx.doi.org/10.21227/fw8e-z983.
2. Amazon (2020). Humor Detection from Product Question Answering Systems [Dataset]. https://registry.opendata.aws/humor-detection/
3. Amaan Mansuri (2023). Humor Detection [Dataset]. https://www.kaggle.com/datasets/amaanmansuri/humor-detection
4. Christ, Lukas; Amiriparian, Shahin; Müller, Niklas; König, Andreas; Schuller, Björn (2022). MuSe-Humor: Humor Detection Sub-Challenge (MuSe 2022) [Dataset]. http://doi.org/10.5281/zenodo.6433326
5. Deep Contractor (2022). 200K SHORT TEXTS FOR HUMOR DETECTION [Dataset]. https://www.kaggle.com/datasets/deepcontractor/200k-short-texts-for-humor-detection
6. Fraser Greenlee (2023). short-jokes [Dataset]. https://huggingface.co/datasets/Fraser/short-jokes
7. Mika Hämäläinen (2020). Movie Title Puns [Dataset]. https://www.kaggle.com/mikahama/movie-title-puns
8. Md. Kamrul Hasan; Wasifur Rahman; Amir Zadeh; Jianyuan Zhong; Md. Iftekhar Tanveer; Louis-Philippe Morency; Mohammed; Hoque, UR-FUNNY Dataset [Dataset]. https://paperswithcode.com/dataset/ur-funny
9. Taiki Kokubun; Satoru Tsuda; Hiroshi Kunikata; Masayuki Yasuda; Noriko Himori; Shiho Kunimatsu-Sanuki; Kazuichi Maruyama; Toru Nakazawa (2018). Characteristic Profiles of Inflammatory Cytokines in the Aqueous Humor of Glaucomatous Eyes [Dataset]. http://doi.org/10.6084/m9.figshare.5116186.v1

10. Kenneth Y. Goldberg; Theresa Roeder; Dhruv Gupta; Chris Perkins, Jester (Jokes) Dataset [Dataset]. https://paperswithcode.com/dataset/jester
11. William Scott (2020). Memotion Dataset 7k [Dataset]. https://www.kaggle.com/williamscott701/memotion-dataset-7k
12. Sampath Nikhalashree; Ronnie George; Balekudaru Shantha; Vijaya Lingam; Wadke Vidya; Manish Panday; Konerirajapuram Natarajan Sulochana; Karunakaran Coral (2023). Detection of Proteins Associated with Extracellular Matrix Regulation in the Aqueous Humour of Patients with Primary Glaucoma [Dataset]. http://doi.org/10.6084/m9.figshare.8094566.v1

Books and References

1. Your Wit Is My Command - Building AIs with a Sense of Humor | By Tony Veale https://mitpress.mit.edu/9780262045995/your-wit-is-my-command/
2. An Introduction to the Psychology of Humor - By Janet M. Gibson https://www.routledge.com/An-Introduction-to-the-Psychology-of-Humor/Gibson/p/book/9780367029081
3. Inside Jokes: Using Humor to Reverse-Engineer the Mind by Matthew Hurley, Jesse Prinz, and Stephen M. Kosslyn https://www.amazon.in/Inside-Jokes-Using-Humor-Reverse-Engineer/dp/0262518694
4. The Humor Code: A Scientific Look at Why Jokes Make Us Laugh by Peter McGraw and Joel Warner https://www.amazon.in/Humor-Code-Global-Search-Things/dp/1451665415
5. The Comic Toolbox: How to be Funny Even if You're Not by John Vorhaus https://www.amazon.com/Comic-Toolbox-Funny-Even-Youre-ebook/dp/B00R3KEMMC
6. Comic Relief: A Comprehensive Philosophy of Humor by John Morreall https://www.amazon.ca/Comic-Relief-Comprehensive-Philosophy-Humor/dp/1405196122