Natural Language Processing

DATS 6312

Fall 2022

Text Classification & Text Summarization

Instructor: Dr. Amir Jafari

Submitted By: Nusrat Nawshin

Submission Date: December 12, 2022

Contents

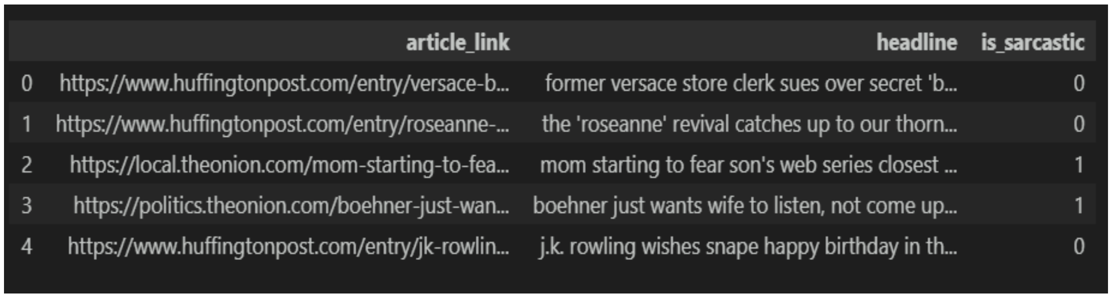
Overview

The goal of our project is to classify sarcastic news headlines and summarize news articles to generate sarcastic headlines in PyTorch.

Introduction and Data Source

The dataset that we are using is collected from Kaggle, News Headlines Dataset For Sarcasm Detection. There are a total of 55,328 news headlines with articles in a json format. Past studies in Sarcasm Detection mostly make use of Twitter datasets collected using hashtag-based supervision but such datasets are noisy in terms of labels and language. To overcome the limitations related to noise in Twitter datasets, this News Headlines dataset for Sarcasm Detection is collected from two news website. *TheOnion* aims at producing sarcastic versions of current events and we collected all the headlines from News in Brief and News in Photos categories (which are sarcastic). We collect real (and non-sarcastic) news headlines from *HuffPost*.[1] This dataset consists of three attributes:

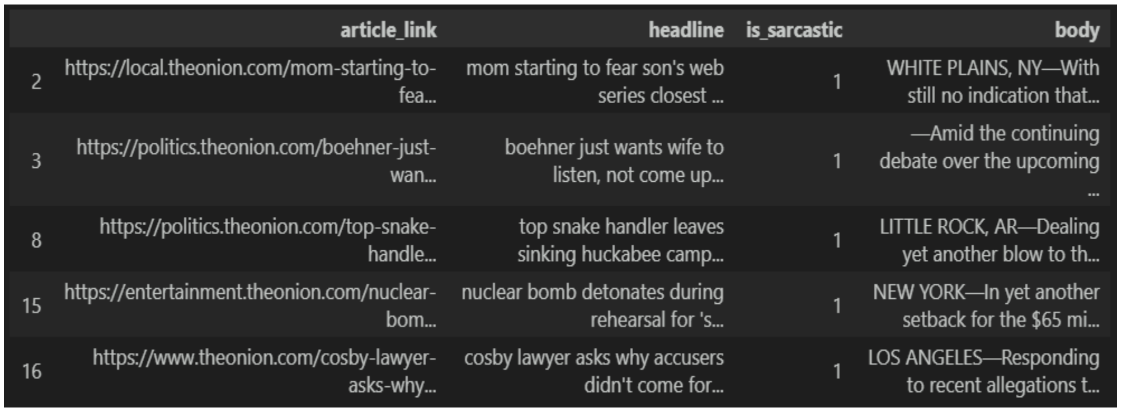
- **is_sarcastic**: 1 if the record is sarcastic otherwise 0
- **headline**: the headline of the news article
- **article_link**: link to the original news article



	article_link	headline	is_sarcastic
0	https://www.huffingtonpost.com/entry/versace-b...	former versace store clerk sues over secret 'b...	0
1	https://www.huffingtonpost.com/entry/roseanne-...	the 'roseanne' revival catches up to our thorn...	0
2	https://local.theonion.com/mom-starting-to-fea...	mom starting to fear son's web series closest ...	1
3	https://politics.theonion.com/boehner-just-wan...	boehner just wants wife to listen, not come up...	1
4	https://www.huffingtonpost.com/entry/jk-rowlin...	j.k. rowling wishes snape happy birthday in th...	0

Figure 1: Original dataset

For summarization, we scraped the article bodies from the internet using the BeautifulSoup python package. The final dataset is in the cloud, so the process does not need to be repeated multiple times.



	article_link	headline	is_sarcastic	body
2	https://local.theonion.com/mom-starting-to-fea...	mom starting to fear son's web series closest ...	1	WHITE PLAINS, NY—With still no indication that...
3	https://politics.theonion.com/boehner-just-wan...	boehner just wants wife to listen, not come up...	1	—Amid the continuing debate over the upcoming ...
8	https://politics.theonion.com/top-snake-handle...	top snake handler leaves sinking huckabee camp...	1	LITTLE ROCK, AR—Dealing yet another blow to th...
15	https://entertainment.theonion.com/nuclear-bom...	nuclear bomb detonates during rehearsal for 's...	1	NEW YORK—In yet another setback for the $65 mi...
16	https://www.theonion.com/cosby-lawyer-asks-why...	cosby lawyer asks why accusers didn't come for...	1	LOS ANGELES—Responding to recent allegations t...

Figure 2: Web Scrapped dataset

Data Preprocessing

For the classification problem few text-processing were performed including removing the numbers, lowering the characters, removing stop words and lemmatization on the new headlines. For removing numbers and punctuations and other noisy characters from the headlines the following regular expression was used:

```
df['text'] = [re.sub(r'[^a-zA-Z]', ' ', str(sentence)) for sentence in df['text']]
```

The headline characters are then converted to lower-case to reduce the noise.

```
final_df['text'] = [sentence.lower() for sentence in final_df['text']]
```

Stop words are removed from the headlines using the NLTK package to remove low-level information and focus on the relevant tokens.

```
lst = []
for sentence in final_df['text']:
    lst.append([i for i in str(sentence).split() if i not in (stopwords.words('english'))])

final_df['text'] = [' '.join(sent) for sent in lst]
```

Finally, the headline texts were lemmatized.

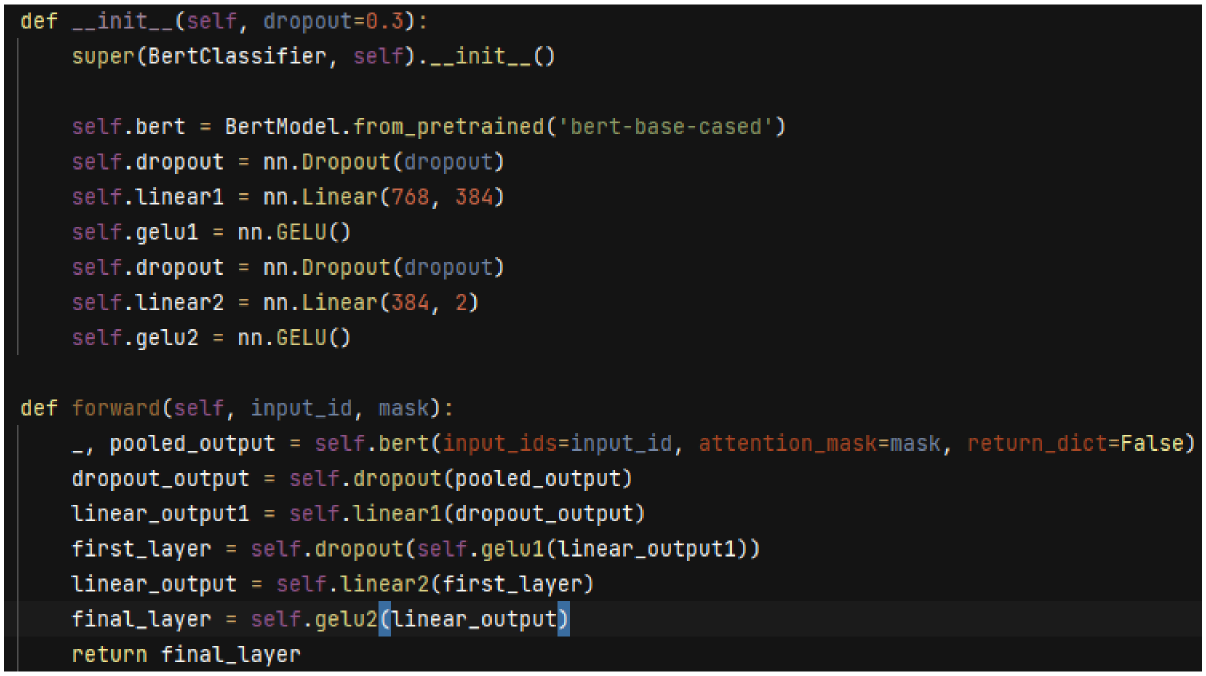
```
wnl = nltk.WordNetLemmatizer()
lst = []
for sentence in final_df['text']:
    lst.append(' '.join(wnl.lemmatize(word) for word in sentence.split()))

final_df['text'] = lst
```

It is a text normalization technique that converts words to their root mode. The context of the text may be helpful in our classification problem. Lemmatization performs better than stemming the word as it has higher accuracy at getting the root of the word. We used the NLTK WordNetLemmatizer method for lemmatization.

Classification – BERT + MLP

BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection i.e., attention. [2] BERT base has 12 layers in the encoder with 768 hidden feedforward layers and 110 million parameters. This model takes CLS token as input first, then it is followed by a sequence of words as input. Here 'CLS' is a classification token. It then passes the input to the above layers. [3] Each layer applies self-attention, passes the result through a feedforward network after then it hands off to the next encoder. The model outputs a vector of hidden size 768 which is then passed in two liner layers with GELU as activation function. For the optimization AdamW optimizer is used with learning rate 1e-6 with ‘CrossEntropyLoss’ as loss function as the target is binary. AdamW is a stochastic optimization method that modifies the typical implementation of weight decay in Adam, by decoupling weight decay from the gradient update.[4]. The model has total 108606338 trainable parameters.

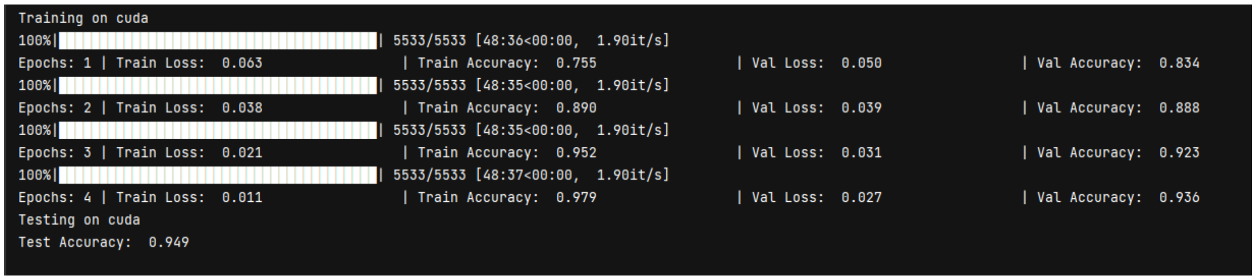
Model:



```
def __init__(self, dropout=0.3):
    super(BertClassifier, self).__init__()

    self.bert = BertModel.from_pretrained('bert-base-cased')
    self.dropout = nn.Dropout(dropout)
    self.linear1 = nn.Linear(768, 384)
    self.gelu1 = nn.GELU()
    self.dropout = nn.Dropout(dropout)
    self.linear2 = nn.Linear(384, 2)
    self.gelu2 = nn.GELU()

def forward(self, input_id, mask):
    _, pooled_output = self.bert(input_ids=input_id, attention_mask=mask, return_dict=False)
    dropout_output = self.dropout(pooled_output)
    linear_output1 = self.linear1(dropout_output)
    first_layer = self.dropout(self.gelu1(linear_output1))
    linear_output = self.linear2(first_layer)
    final_layer = self.gelu2(linear_output)
    return final_layer
```

Figure 3: BERT-MLP classification model

After training it with batch size 8 and till 4 epochs, the training accuracy is 0.97 and test accuracy is 0.94.



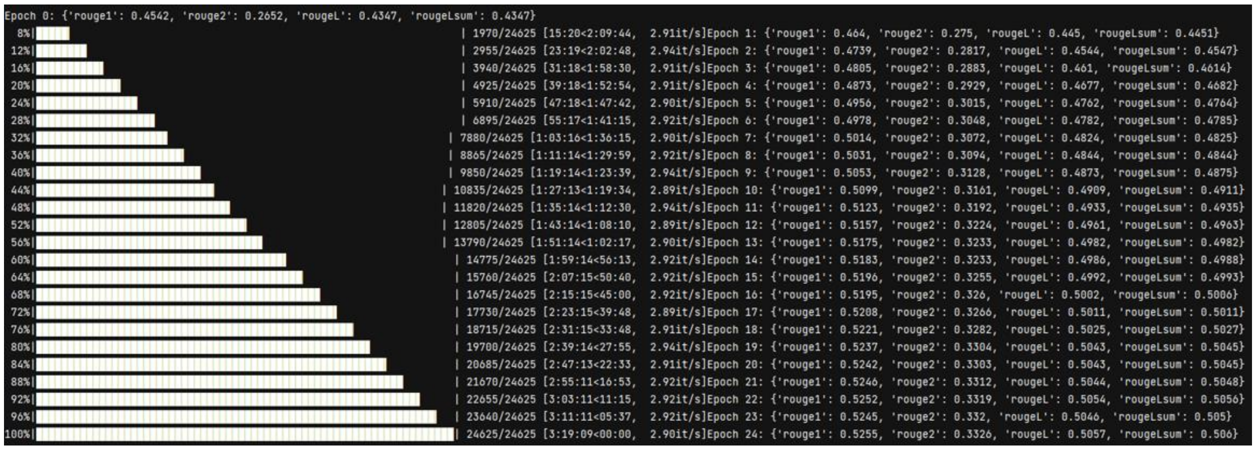
```
Training on cuda
100%|██████████████████████████████| 5533/5533 [48:36<00:00,  1.90it/s]
Epochs: 1 | Train Loss:  0.063          | Train Accuracy:  0.755          | Val Loss:  0.050          | Val Accuracy:  0.834
100%|██████████████████████████████| 5533/5533 [48:35<00:00,  1.90it/s]
Epochs: 2 | Train Loss:  0.038          | Train Accuracy:  0.890          | Val Loss:  0.039          | Val Accuracy:  0.888
100%|██████████████████████████████| 5533/5533 [48:35<00:00,  1.90it/s]
Epochs: 3 | Train Loss:  0.021          | Train Accuracy:  0.952          | Val Loss:  0.031          | Val Accuracy:  0.923
100%|██████████████████████████████| 5533/5533 [48:37<00:00,  1.90it/s]
Epochs: 4 | Train Loss:  0.011          | Train Accuracy:  0.979          | Val Loss:  0.027          | Val Accuracy:  0.936
Testing on cuda
Test Accuracy:  0.949
```

Figure 4: BERT-MLP classification model output

Summarization – T5-small-headline-generator

For summarization on the articles with sarcastic headlines are used. It has 1970 articles.

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. The input sequence is fed to the model using input_ids. The target sequence is shifted to the right, i.e., prepended by a start-sequence token and fed to the decoder using the decoder_input_ids. In teacher-forcing style, the target sequence is then appended by the EOS token and corresponds to the labels. The PAD token is hereby used as the start-sequence token. [5] For this summarization, T5-small-headline-generator model is used, which is t5-small fine-tuned for headline generation. The articles of sarcastic news are passed as the input of the summarization which is summarizing and giving the news headlines. The batch size is 16 and it’s trained till 25 epochs. AdamW optimizer is used with learning rate 2e-5.



```
Epoch 0: {'rouge1': 0.4542, 'rouge2': 0.2652, 'rougeL': 0.4347, 'rougeLsum': 0.4347}
  8%|██                       | 1970/24625 [15:20<2:09:44,  2.91it/s]Epoch 1: {'rouge1': 0.464, 'rouge2': 0.275, 'rougeL': 0.445, 'rougeLsum': 0.4451}
 12%|███                      | 2955/24625 [23:19<2:02:48,  2.94it/s]Epoch 2: {'rouge1': 0.4739, 'rouge2': 0.2817, 'rougeL': 0.4544, 'rougeLsum': 0.4547}
 16%|████                     | 3940/24625 [31:18<1:58:30,  2.91it/s]Epoch 3: {'rouge1': 0.4805, 'rouge2': 0.2883, 'rougeL': 0.461, 'rougeLsum': 0.4614}
 20%|█████                    | 4925/24625 [39:18<1:52:54,  2.91it/s]Epoch 4: {'rouge1': 0.4873, 'rouge2': 0.2929, 'rougeL': 0.4677, 'rougeLsum': 0.4682}
 24%|██████                   | 5910/24625 [47:18<1:47:42,  2.90it/s]Epoch 5: {'rouge1': 0.4956, 'rouge2': 0.3015, 'rougeL': 0.4762, 'rougeLsum': 0.4764}
 28%|███████                  | 6895/24625 [55:17<1:41:15,  2.92it/s]Epoch 6: {'rouge1': 0.4978, 'rouge2': 0.3048, 'rougeL': 0.4782, 'rougeLsum': 0.4785}
 32%|████████                 | 7880/24625 [1:03:16<1:36:15,  2.90it/s]Epoch 7: {'rouge1': 0.5014, 'rouge2': 0.3072, 'rougeL': 0.4824, 'rougeLsum': 0.4825}
 36%|█████████                | 8865/24625 [1:11:14<1:29:59,  2.92it/s]Epoch 8: {'rouge1': 0.5031, 'rouge2': 0.3094, 'rougeL': 0.4844, 'rougeLsum': 0.4844}
 40%|██████████               | 9850/24625 [1:19:14<1:23:39,  2.94it/s]Epoch 9: {'rouge1': 0.5053, 'rouge2': 0.3128, 'rougeL': 0.4873, 'rougeLsum': 0.4875}
 44%|███████████              | 10835/24625 [1:27:13<1:19:34,  2.89it/s]Epoch 10: {'rouge1': 0.5099, 'rouge2': 0.3161, 'rougeL': 0.4909, 'rougeLsum': 0.4911}
 48%|████████████             | 11820/24625 [1:35:14<1:12:30,  2.94it/s]Epoch 11: {'rouge1': 0.5123, 'rouge2': 0.3192, 'rougeL': 0.4933, 'rougeLsum': 0.4935}
 52%|█████████████            | 12805/24625 [1:43:14<1:08:10,  2.89it/s]Epoch 12: {'rouge1': 0.5157, 'rouge2': 0.3224, 'rougeL': 0.4961, 'rougeLsum': 0.4963}
 56%|██████████████           | 13790/24625 [1:51:14<1:02:17,  2.90it/s]Epoch 13: {'rouge1': 0.5175, 'rouge2': 0.3233, 'rougeL': 0.4982, 'rougeLsum': 0.4982}
 60%|███████████████          | 14775/24625 [1:59:14<56:13,  2.92it/s]Epoch 14: {'rouge1': 0.5183, 'rouge2': 0.3233, 'rougeL': 0.4986, 'rougeLsum': 0.4988}
 64%|████████████████         | 15760/24625 [2:07:15<50:40,  2.92it/s]Epoch 15: {'rouge1': 0.5196, 'rouge2': 0.3255, 'rougeL': 0.4992, 'rougeLsum': 0.4993}
 68%|█████████████████        | 16745/24625 [2:15:15<45:00,  2.92it/s]Epoch 16: {'rouge1': 0.5195, 'rouge2': 0.326, 'rougeL': 0.5002, 'rougeLsum': 0.5006}
 72%|██████████████████       | 17730/24625 [2:23:15<39:48,  2.89it/s]Epoch 17: {'rouge1': 0.5208, 'rouge2': 0.3266, 'rougeL': 0.5011, 'rougeLsum': 0.5011}
 76%|███████████████████      | 18715/24625 [2:31:15<33:48,  2.91it/s]Epoch 18: {'rouge1': 0.5221, 'rouge2': 0.3282, 'rougeL': 0.5025, 'rougeLsum': 0.5027}
 80%|████████████████████     | 19700/24625 [2:39:14<27:55,  2.94it/s]Epoch 19: {'rouge1': 0.5237, 'rouge2': 0.3304, 'rougeL': 0.5043, 'rougeLsum': 0.5045}
 84%|█████████████████████    | 20685/24625 [2:47:13<22:33,  2.91it/s]Epoch 20: {'rouge1': 0.5242, 'rouge2': 0.3303, 'rougeL': 0.5043, 'rougeLsum': 0.5045}
 88%|██████████████████████   | 21670/24625 [2:55:11<16:53,  2.92it/s]Epoch 21: {'rouge1': 0.5246, 'rouge2': 0.3312, 'rougeL': 0.5044, 'rougeLsum': 0.5048}
 92%|███████████████████████  | 22655/24625 [3:03:11<11:15,  2.92it/s]Epoch 22: {'rouge1': 0.5252, 'rouge2': 0.3319, 'rougeL': 0.5054, 'rougeLsum': 0.5056}
 96%|████████████████████████ | 23640/24625 [3:11:11<05:37,  2.92it/s]Epoch 23: {'rouge1': 0.5245, 'rouge2': 0.332, 'rougeL': 0.5046, 'rougeLsum': 0.505}
100%|█████████████████████████| 24625/24625 [3:19:09<00:00,  2.90it/s]Epoch 24: {'rouge1': 0.5255, 'rouge2': 0.3326, 'rougeL': 0.5057, 'rougeLsum': 0.506}
```

Figure 5: Summarization T5-small-headline-generator scores

The summarization gives all ROUGE scores more than 50. ROUGH stands for Recall-Oriented Understudy for Gisting Evaluation which is essentially a set of metrics for evaluating automatic summarization of texts. [6] In this task it works by comparing the automatically produced summary against a set of reference summaries i.e., the news headlines. As our target headlines are just one sentence, we use lead-1 baseline to set the baseline scores instead of lead-3. It returns the first line of the text as the summary.

Baseline Scores on validation set:

Rouge1 : 0.305	RougeL : 0.280
Rouge2 : 0.151	RougeLsum : 0.280

Final ROUGE scores:

Rouge1 = 0.5255	RougeL = 0.5057
Rouge2 = 0.3326	RougeLsum = 0.506

Summarized Outputs:

Example 1:

Article: SAN FRANCISCO—In an effort to stop the spread of the dangerous negativity as quickly as possible, agents from the Department of Labor’s Emergency Response Team on Friday reportedly sealed off the toxic workplace environment at tech incubator Molloy Capital. We’ve begun the hazardous but absolutely necessary task of cleaning up the site’s dangerous backbiting, unwanted sexual advances, and gross managerial incompetence, said Labor Department special agent Angela Lasker, who noted that prior to their arrival, a steady stream of inappropriate comments had been seeping into the water-cooler conversation and were threatening to poison the entire company-wide intranet. While we believe we’ve contained most of the professional and psychological damage, it’s still a very hostile, unsafe area. We honestly don’t think it will be able to sustain new hires for the foreseeable future, or at least until the pervasive low morale has had a chance to dissipate. Lasker went on to say, however, that the communications department would likely never again be suitable for human employment.

Headline: department of labor response team seals off toxic workplace environment

Summarized Headline: department of labor seals off toxic workplace environment

Example 2:

Article: DENTON, TX—Stressing that they were there solely to purchase gasoline and use the bathroom, if necessary, area dad Mike Whitcomb clarified while pulling into a travel plaza Thursday that this was not a food stop. We’re here to get gas and that’s it, Whitcomb said emphatically, adding that his three children were welcome to get out and stretch their legs, but they had better be back in their seats and buckled up by the time he finished filling the tank because he wasn’t waiting around. I want to be back on the road in five minutes. If you’re hungry, you can have one of the apples your mom brought. Sources later confirmed area mother Debra Whitcomb had okayed one bag of Chex Mix for everyone to share.

Headline: dad clarifies this not a food stop

Summarized Headline: area dad clarifies this not a food stop

Example 3:

Article: EVANSVILLE, IN—Describing the unholy intermixture of letters and numbers as repulsive and utterly vile, disgusted sources confirmed Monday that the Netflix password of local parents Evan and Jeannine Perkins was a nauseatingly grotesque combination of their children’s names and birthdays. The 19-character code, which swaps out an S in their firstborn’s name with a hideous and debasing dollar sign, reportedly alternates at points between upper- and lowercase letters, undulating back and forth in a sickening chain that renders the names Sophia and Ben wholly unrecognizable. Further reports indicate that after a perverse fusing of the digits in the children’s birth years, the stomach-turning amalgamation terminates with the loathsome and unnatural conjoinment of the family dog’s nickname, mercifully bringing to an end a monstrosity so hideous, it reportedly can only be gazed upon with the characters replaced by a series of concealing asterisks. Sources confirmed that if anything positive could be stated about the unutterable atrocity of a password, it was that it was strong.

Headline: parents' password a grotesque combination of children's names, birthdays

Summarized Headline: internet password nauseatingly grotesque combination of children's names, birthdays

As we can see the summarized headlines and the actual sarcastic headlines are quite similar.

Conclusion

The news article’s sarcastic headline classification model using pretrained BERT followed by MLP is performing very accurately with 97% training accuracy and 94% testing accuracy. In the text summarization, the model is performing pretty well the headlines are almost very close and the ROUGH scores of more than 50%. Limitation on the computing power restricted our attempts on better model building by modeling on higher hyperparameters like batch size, optimizers and bigger transfer models.

Percentage of Code Written

The codes were referred from Hugging Face official sites and documentations. Approximately 65% of the code was referred from the internet.

References

[1] https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection

[2] https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model

[3] https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/

[4] https://paperswithcode.com/method/adamw

[5] https://huggingface.co/docs/transformers/model_doc/t5

[6] An intro to ROUGE, and how to use it to evaluate summaries (freecodecamp.org)