

# Assignment 4

## Text Pre-Processing

```
def expand_acronyms(text):
    expanded_words = []
    for word in text.split():
        expanded_words.append(contractions.fix(word))
    expanded_text = ' '.join(expanded_words)
    return expanded_text

def remove_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

def remove_accented_chars(text):
    return unicode(text)

def normalize_text(text):
    text = str(text)
    text = text.lower()
    text = expand_acronyms(text)
    text = remove_html_tags(text)
    text = remove_accented_chars(text)
    text = re.sub('\s\\W', ' ', text)
    text = re.sub(['\\W,\\s', ' ', text])
    text = re.sub("\\d+", "", text)
    text = re.sub('\\s+', ' ', text)
    return text
```

## Data Split

```
training_data = []
testing_data = []
training_reviews = []
summary_token = ' TL;DR '
```

  

```
for id in range(int(len(dataset)*0.75)):
    # print(id,"/",int(len(dataset)*0.75))
    text = normalize_text(dataset['Text'][id])
    summary = normalize_text(dataset['Summary'][id])
    training_data.append([text, summary])
    training_reviews.append(text + summary_token + summary)
```

  

```
for id in range(int(len(dataset)*0.75), len(dataset)):
    # print(id - int(len(dataset)*0.75),"/",int(len(dataset)*0.25))
    text = normalize_text(dataset['Text'][id])
    summary = normalize_text(dataset['Summary'][id])
    testing_data.append([text, summary])
```

I have taken 15000 samples due to computational issues.

## Custom Dataset

```
class SummaryData(Dataset):
    def __init__(self, data, tokenizer):
        self.data = data
        tokenizer.pad_token = '[PAD]'
        self.X_encoded = tokenizer(self.data, max_length=100, truncation=True, padding="max_length", return_tensors="pt")
        self.input_ids = self.X_encoded['input_ids']
        self.attention_mask = self.X_encoded['attention_mask']

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return (self.input_ids[idx], self.attention_mask[idx])
```

## Loading Pre-trained Model

```
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")
```

## Fine-tuning

```
def train(data_loader, model, optim):  
  
    epochs = 10  
  
    for i in range(epochs):  
        train_loss = 0  
        for input_ids, att_mask in tqdm.tqdm(data_loader):  
            input_ids = input_ids.to(device)  
            att_mask = att_mask.to(device)  
            optim.zero_grad()  
            loss = model(input_ids, attention_mask=att_mask, labels=input_ids).loss  
            train_loss += loss  
            loss.backward()  
            optim.step()  
            torch.cuda.empty_cache()  
  
        print("Epoch: ", i, " train_loss", loss)
```

## Final Parameters

1. Batch size - 32
2. Learning Rate - 0.0003
3. Optimizer - Adam
4. Number of epochs - 10

## Rogue Scores

```
{'rouge-1': {'r': 0.07132360010360002, 'p': 0.0917623879823879, 'f': 0.07307514845082086},  
'rouge-2': {'r': 0.012565935545935547, 'p': 0.014927532467532475, 'f':  
0.011839238116829786}, 'rouge-l': {'r': 0.07022698634698622, 'p': 0.09052841972841967, 'f':  
0.07198937844740379}}
```