

# Chandreen\_1158931\_Project\_W2023

April 16, 2023

## 1 How Data Augmentation using Back-Translation Affects the Text Classification Performance

Student Name: Chandreen Ravihari Liyanage

Student ID: 1158931

### *Main aim*

Description about the project: The main aim of this project to evaluate how the performance of text classification task will be affected by augmenting data using “backtransfer” technique. Backtransfer is a technique, which convert the original text in one language (here English) into another language (here French) and again transfer from second language to the first language. This technique has proposed that during this transformation, the original text will be re-generated differently, so that they can be used as augmented text.

### *The method*

Hence, in this work I used this backtranslation technique to balance the unbalanced dataset; where I had number of records before (0-159, 1-219, 2-109), after (0 - 219, 1 -219, 2 -218). Then I used the BERT model (bert-base-uncased) and finetuned separately on the original and augmented dataset and tested the performance using a separate non-augmented dataset.

### *Hyper-parameters*

The hyper-parameter setup used in this evaluation is , epochs = 10, batch size=16, learning rate =  $2e-5$ , optimizer= Adam, and epsilon= $1e-8$ .

### *Dataset*

This Twitter dataset is related to abortion stance classification and has three classes, 0-against, 1-favor and 2-none considering the stance target ” legalization of abortion“. The dataset is taken from”<https://paperswithcode.com/dataset/tweeteval>”

### *Preprocessing*

I performed preprocessing the text at two stages, 1) before augmenting data and 2) before classification. The stage 1 techniques are simple, such as removal of #, @user, urls, additional spaces, line breaks and punctuations. The reason for performing simple techniques is to keep the augmented data more similar to the original text. However, during stage 2 of preprocessing, I performed stopword removal, stemming and lemmatization.

### **Conclusion**

According to the results the testing accuracy has been reduced from 62% to 35% in the presence of the augmented dataset. And also, the backtranslation technique has generated many duplicated records. Duplicate records with consistent labels essentially provide redundant information, as they do not add any new information to the model. In fact, including duplicate records in such cases may introduce bias, as the model may become overly reliant on the duplicated data and overestimate the importance of certain patterns, leading to overfitting. However, we could not see any overfitting in the results.

```
[ ]: !pip install transformers
      !pip install sentencepiece
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
```

```
Collecting transformers
```

```
  Downloading transformers-4.28.1-py3-none-any.whl (7.0 MB)
```

```
7.0/7.0 MB
```

```
59.4 MB/s eta 0:00:00
```

```
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-
packages (from transformers) (3.11.0)
```

```
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-
packages (from transformers) (6.0)
```

```
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.9/dist-packages (from transformers) (2022.10.31)
```

```
Collecting huggingface-hub<1.0,>=0.11.0
```

```
  Downloading huggingface_hub-0.13.4-py3-none-any.whl (200 kB)
```

```
200.1/200.1 kB
```

```
10.0 MB/s eta 0:00:00
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from transformers) (2.27.1)
```

```
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-
packages (from transformers) (23.0)
```

```
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.9/dist-
packages (from transformers) (4.65.0)
```

```
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
```

```
  Downloading
```

```
tokenizers-0.13.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8
MB)
```

```
7.8/7.8 MB
```

```
68.6 MB/s eta 0:00:00
```

```
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.9/dist-packages (from transformers) (1.22.4)
```

```
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.9/dist-packages (from huggingface-
hub<1.0,>=0.11.0->transformers) (4.5.0)
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->transformers) (3.4)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->transformers) (2022.12.7)
```

```
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->transformers) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->transformers) (1.26.15)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.13.4 tokenizers-0.13.3
transformers-4.28.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting sentencepiece
  Downloading
sentencepiece-0.1.98-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(1.3 MB)
1.3/1.3 MB
55.0 MB/s eta 0:00:00
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.98
```

```
[ ]: import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
```

```
[ ]: nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] /root/nltk_data...
[nltk_data] Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

```
[ ]: True
```

```
[ ]: import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader, RandomSampler,
SequentialSampler, random_split
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
import transformers
from transformers import BertForSequenceClassification, AdamW, BertConfig, BertTokenizer, get_linear_schedule_with_warmup
```

```
[ ]: import random
import time
import datetime
import gc
```

```
[ ]: import pandas as pd
import json
import csv
import numpy as np

import os
import time
import re
```

```
[ ]: from transformers import MarianMTModel, MarianTokenizer
```

```
[ ]: from google.colab import drive
drive.mount('/content/gdrive')
```

Mounted at /content/gdrive

```
[ ]: # Setting the base path
base_path = "/content/gdrive/MyDrive/Lakehead//NLP_Class/Project"
```

```
[ ]: os.chdir(base_path)
```

```
[ ]: df = pd.read_csv("TweetEval.csv")
df
```

```
[ ]: Unnamed: 0      text  label \
0      0  we remind ourselves that love means to be will...    1
1      1  @user @user and most Islanders have different ...    0
2      2  Life is #precious & so are babies, mothers, & ...    1
3      3  @user too many people are taking this to serio...    0
4      4  Dude i won a #freeshirt from @user ! I never w...    0
..      ...      ...      ...
482     578  Oh look!!! So not only are antichoice strongly...    2
483     579  @user I don't think I have a right to use some...    2
484     581  People aren't 'pro-life' they're 'pro-birth'. ...    2
485     582  @user i don't follow the news, is there a new ...    0
486     583  @user The Gods #Law, the #Truth, is being #sup...    0
```

label\_text

```

0      against
1      none
2      against
3      none
4      none
..     ...
482    favor
483    favor
484    favor
485    none
486    none

```

[487 rows x 4 columns]

```

[ ]: # I kept a separate dataset to evaluate the classification performance before
      ↪and after data augmentation
df_test = pd.read_csv("TweetEval_testdata.csv")
df_test

```

```

[ ]:      Unnamed: 0      text  label  \
0          0  Need a ProLife R.E. Agent? - Support a ProLife...      1
1          1  Where is the childcare program @user which you...      1
2          2  I get several requests with petitions to save ...      1
3          3  we must always see others as Christ sees us,we...      1
4          4  PRAYERS FOR BABIES Urgent prayer one in Lexing...      1
..         ...
176        336  Every time you respond to something that frust...      0
177        337  Obamcare loses again in the Supreme Court. Rel...      0
178        342  BRAVO TO BOTH OF YOU BEAUTIFULLY EXPLAINED HOL...      2
179        344  My body, my life. You fuck it up in a way I'm ...      2
180        345  Thank you @user for treating me with kindness ...      0

```

```

      label_text
0      against
1      against
2      against
3      against
4      against
..     ...
176    none
177    none
178    favor
179    favor
180    none

```

[181 rows x 4 columns]

```
[ ]: device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
device
```

```
[ ]: device(type='cuda', index=0)
```

## 1.1 Recording class counts

```
[ ]: df['label'].value_counts()
```

```
[ ]: 1    219
      0    159
      2    109
      Name: label, dtype: int64
```

```
[ ]: class_counts = df['label'].value_counts().reset_index()
      class_counts.columns = ["class", "count"]
      class_counts
```

```
[ ]:      class  count
      0      1    219
      1      0    159
      2      2    109
```

```
[ ]: c1=class_counts.loc[class_counts['class'] == 0, 'count']
      c2=class_counts.loc[class_counts['class'] == 1, 'count']
      c3=class_counts.loc[class_counts['class'] == 2, 'count']

      c1=int(c1)
      c2=int(c2)
      c3=int(c3)
```

```
[ ]: df_test['label'].value_counts()
```

```
[ ]: 0    63
      1    60
      2    58
      Name: label, dtype: int64
```

## 1.2 Preprocessing

```
[ ]: # Perform some preprocessing relevant to Twitter text. I do not perform
      ↳ stemming, lemmatization as we need text
      # more similar to the original text since we are performing data augmentation

      def clean_text(text: str):
```

```

text = text.lower() #change case to lower
text = text.strip() # Remove extra spaces
text = text.replace("#", "") # Remove hashtags
text = text.replace("@user", "") # Remove hashtags
text = re.sub(r"http\S+", "", text) # Removed urls
text = re.sub(r"[^\w\s]", "", text) # Removed punctuations #Sentence
return text

```

```
[ ]: df["text"] = df.text.apply(lambda x: clean_text(x))
```

```
[ ]: df
```

```
[ ]:
      Unnamed: 0      text  label \
0          0  we remind ourselves that love means to be will...    1
1          1    and most islanders have different definition...    0
2          2  life is precious so are babies mothers fathe...    1
3          3  too many people are taking this to seriously ...    0
4          4  dude i won a freeshirt from i never win anyt...    0
..          ...
482        578  oh look so not only are antichoice strongly ag...    2
483        579  i dont think i have a right to use someone el...    2
484        581  people arent prolife theyre probirth when a mo...    2
485        582  i dont follow the news is there a new law tha...    0
486        583  the gods law the truth is being supressed the...    0

      label_text
0      against
1         none
2      against
3         none
4         none
..          ...
482      favor
483      favor
484      favor
485         none
486         none

```

```
[487 rows x 4 columns]
```

### 1.3 Data Augmentation using Backtranslation

Using Backtranslating technique, I convert the selected English text into french and then convert it to English to use as augmented data.

the first model translating English to French

```
[ ]: # Get the name of the first model
first_model_name = 'Helsinki-NLP/opus-mt-en-fr'

# Get the tokenizer
first_model_tkn = MarianTokenizer.from_pretrained(first_model_name)

# Load the pretrained model based on the name
first_model = MarianMTModel.from_pretrained(first_model_name)
```

```
Downloading (...)olve/main/source.spm: 0%|          | 0.00/778k [00:00<?, ?B/s]
Downloading (...)olve/main/target.spm: 0%|          | 0.00/802k [00:00<?, ?B/s]
Downloading (...)olve/main/vocab.json: 0%|          | 0.00/1.34M [00:00<?, ?B/s]
Downloading (...)okenizer_config.json: 0%|          | 0.00/42.0 [00:00<?, ?B/s]
Downloading (...)olve/main/config.json: 0%|          | 0.00/1.42k [00:00<?, ?B/s]
/usr/local/lib/python3.9/dist-
packages/transformers/models/arian/tokenization_arian.py:194: UserWarning:
Recommended: pip install sacremoses.
  warnings.warn("Recommended: pip install sacremoses.")
Downloading pytorch_model.bin: 0%|          | 0.00/301M [00:00<?, ?B/s]
Downloading (...)neration_config.json: 0%|          | 0.00/293 [00:00<?, ?B/s]
the second model translating French to English.
```

```
[ ]: # Get the name of the second model
second_model_name = 'Helsinki-NLP/opus-mt-fr-en'

# Get the tokenizer
second_model_tkn = MarianTokenizer.from_pretrained(second_model_name)

# Load the pretrained model based on the name
second_model = MarianMTModel.from_pretrained(second_model_name)
```

```
Downloading (...)olve/main/source.spm: 0%|          | 0.00/802k [00:00<?, ?B/s]
Downloading (...)olve/main/target.spm: 0%|          | 0.00/778k [00:00<?, ?B/s]
Downloading (...)olve/main/vocab.json: 0%|          | 0.00/1.34M [00:00<?, ?B/s]
Downloading (...)okenizer_config.json: 0%|          | 0.00/42.0 [00:00<?, ?B/s]
Downloading (...)olve/main/config.json: 0%|          | 0.00/1.42k [00:00<?, ?B/s]
Downloading pytorch_model.bin: 0%|          | 0.00/301M [00:00<?, ?B/s]
Downloading (...)neration_config.json: 0%|          | 0.00/293 [00:00<?, ?B/s]
```

to properly implement the translation feature, we need to add the special token `>>{tgt}<<` in front of each text that needs to be translated. `{tgt}` is either `fr` or `en`.



```
[ ]: def format_batch_texts(language_code, batch_texts):

    formatted_batch = ">>{}<< {}".format(language_code, batch_texts)

    return formatted_batch
```

function responsible for the translation of the batch of texts

```
[ ]: def perform_translation(batch_texts, model, tokenizer, language="fr"):
    # Prepare the text data into appropriate format for the model
    formatted_batch_texts = format_batch_texts(language, batch_texts)

    # Generate translation using model
    translated = model.generate(**tokenizer(formatted_batch_texts,
    ↪return_tensors="pt", padding=True))

    # Convert the generated tokens indices back into text
    translated_texts = tokenizer.decode(translated[0], skip_special_tokens=True)

    return translated_texts
```

```
[ ]: def perform_back_translation(batch_texts, original_language="en",
    ↪temporary_language="fr"):

    # Translate from Original to Temporary Language
    tmp_translated_batch = perform_translation(batch_texts, first_model,
    ↪first_model_tkn, temporary_language)

    # Translate Back to English
    back_translated_batch = perform_translation(tmp_translated_batch,
    ↪second_model, second_model_tkn, original_language)

    return back_translated_batch
```

```
[ ]: # Declare global lists to store results

augmented_data = []
augmented_data_labels = []
original_data = []
```

```
[ ]: def appending_data(new_text, label, original):

    augmented_data.append(new_text)
    augmented_data_labels.append(label)
    original_data.append(original)
```

```
[ ]: # Call API to generate text

class_1=0
class_2=0
class_3=0

# Set the max records upto 219 as class 1 has this highest number of records
max_records=219
i=0

for index, row in df.iterrows():

    original_text = row['text']
    class_label = row['label']

    print(i)
    i+=1

    # calling the function to add a token to the text
    formatted_text = format_batch_texts("fr", original_text)
    # print(formatted_text)
    # print(formatted_exp)
    # print("\n")

    if(class_label==0 and c1+class_1 < max_records):
        augmented_text = perform_back_translation(formatted_text)
        appending_data(augmented_text, class_label, original_text)
        class_1+=1

    elif (class_label==1 and c2+class_2 < max_records):
        augmented_text = perform_back_translation(formatted_text)
        appending_data(augmented_text, class_label, original_text)
        class_2+=1

    elif (class_label==2 and c3+class_3 < max_records):
        augmented_text = perform_back_translation(formatted_text)
        appending_data(augmented_text, class_label, original_text)
        class_3+=1
```

0

1

/usr/local/lib/python3.9/dist-packages/transformers/generation/utils.py:1313:  
UserWarning: Using `max\_length`'s default (512) to control the generation  
length. This behaviour is deprecated and will be removed from the config in v5  
of Transformers -- we recommend using `max\_new\_tokens` to control the maximum  
length of the generation.

```
warnings.warn(  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47
```

48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95

96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143

144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191

192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239

240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287



288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335

336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383

384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431

432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479

```
480
481
482
483
484
485
486
```

```
[ ]: New_dataset = pd.DataFrame({'original_text': original_data, 'new_text':  
    ↪ augmented_data, 'label': augmented_data_labels})
```

```
[ ]: New_dataset
```

```
[ ]:                                     original_text \
0      and most islanders have different definition...
1      too many people are taking this to seriously ...
2      dude i won a freeshirt from  i never win anyt...
3      like yall can try and push your views on me an...
4      just because it is legal doesnt make it god ho...
..
164    complications come with all types of medical ...
165    what doesnt reduce abortions making it illegal...
166    oh look so not only are antichoice strongly ag...
167    i dont think i have a right to use someone el...
168    people arent prolife theyre probirth when a mo...

                                     new_text  label
0      Most islanders have different definitions of r...      0
1              Too many people take this seriously.      0
2      Dude I won a freeshirt from I never won anythi...      0
3      As Yall can try and push your views on me anyw...      0
4      Just because it's legal doesn't make God the h...      0
..
164    · Complications come with all types of medical...      2
165    What does not reduce abortions makes it illega...      2
166    So look at not only the anti-choice is strongl...      2
167    I don't think I have the right to use someone ...      2
168    • People are not proliferated they are prolife...      2
```

```
[169 rows x 3 columns]
```

```
[ ]: New_dataset['label'].value_counts()
```

```
[ ]: 2    109
      0     60
      Name: label, dtype: int64
```

```
[ ]: New_dataset.to_csv("augmented_data.csv", index=False)
```

```
[ ]: New_dataset = pd.read_csv("augmented_data.csv")
```

#### 1.4 Combine augmented data with original dataset

```
[ ]: # prepare original dataset and testing set
originaldata = df.drop(['Unnamed: 0', 'label_text'], axis=1)
originaldata

testdata = df_test.drop(['Unnamed: 0', 'label_text'], axis=1)
testdata
```

```
[ ]:
```

	text	label
0	Need a ProLife R.E. Agent? - Support a ProLife...	1
1	Where is the childcare program @user which you...	1
2	I get several requests with petitions to save ...	1
3	we must always see others as Christ sees us,we...	1
4	PRAYERS FOR BABIES Urgent prayer one in Lexing...	1
..	...	...
176	Every time you respond to something that frust...	0
177	Obamacare loses again in the Supreme Court. Rel...	0
178	BRAVO TO BOTH OF YOU BEAUTIFULLY EXPLAINED HOL...	2
179	My body, my life. You fuck it up in a way I'm ...	2
180	Thank you @user for treating me with kindness ...	0

[181 rows x 2 columns]

```
[ ]: # prepare augmented dataset
augmented = New_dataset.drop(['original_text'], axis=1)
augmented
```

```
[ ]:
```

	new_text	label
0	Most islanders have different definitions of r...	0
1	Too many people take this seriously.	0
2	Dude I won a freeshirt from I never won anythi...	0
3	As Yall can try and push your views on me anyw...	0
4	Just because it's legal doesn't make God the h...	0
..	...	...
164	• Complications come with all types of medical...	2
165	What does not reduce abortions makes it illeg...	2
166	So look at not only the anti-choice is strongl...	2
167	I don't think I have the right to use someone ...	2
168	• People are not proliferated they are prolife...	2

[169 rows x 2 columns]

```
[ ]: frames = [originaldata, augmented.rename(columns={'new_text':'text'})]
final = pd.concat(frames, ignore_index=True)
final
```

```
[ ]:
      text  label
0    we remind ourselves that love means to be will...    1
1      and most islanders have different definition...    0
2    life is precious so are babies mothers fathe...    1
3    too many people are taking this to seriously ...    0
4    dude i won a freeshirt from i never win anyt...    0
..      ...      ...
651  · Complications come with all types of medical...    2
652  What does not reduce abortions makes it illeg...    2
653  So look at not only the anti-choice is strongl...    2
654  I don't think I have the right to use someone ...    2
655  • People are not proliferated they are prolife...
```

[656 rows x 2 columns]

```
[ ]: final['label'].value_counts()
```

```
[ ]: 1    219
      0    219
      2    218
      Name: label, dtype: int64
```

```
[ ]: final.to_csv("final_dataset.csv", index=False)
```

## 1.5 Preprocessing data for classification

In addition to basic preprocessing, perform further preprocessing, such as stop word removal, stemming, lemmatization.

```
[ ]: # This is the function to map NTLK position tags - for lemmatizing
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
[ ]: # Use a tokenizer from the NLTK library

def preprocessing_stage2(text: str):

    text = text.lower() #change case to lower
    text = text.strip() # Remove extra spaces
    text = text.replace("#", "") # Remove hashtags
    text = text.replace("@user", "") # Remove hashtags
    text = re.sub(r"http\S+", "", text) # Removed urls
    text = re.sub(r"[\w\s]", "", text) # Removed punctuations #Sentence

    processed_sentence = []
    stemmed_sentence = []
    lemmatized_sentence = []

    # First stopwords removal-----
    stopwords = ["that", "and", "is", "it", "a", "an", "the", "this", "to"]

    # Tokenize the sentence
    words = word_tokenize(text)
    for w in words:
        if w not in stopwords:
            processed_sentence.append(w)
    text = " ".join(processed_sentence)

    #second stemming-----
    # Initialize the stemmer
    snow = SnowballStemmer('english')

    # Tokenize the sentence
    words = word_tokenize(text)
    for w in words:
        # Stem the word/token
        stemmed_sentence.append(snow.stem(w))
    stemmed_text = " ".join(stemmed_sentence)

    #third lemmatization-----
    # Initialize the lemmatizer
    wl = WordNetLemmatizer()

    # Tokenize the sentence
    words = word_tokenize(stemmed_text)
    # Get position tags
```



```

word_pos_tags = nltk.pos_tag(words)
# Map the position tag and lemmatize the word or token
for idx, tag in enumerate(word_pos_tags):
    lemmatized_sentence.append(wl.lemmatize(tag[0],
    ↪get_wordnet_pos(tag[1])))

lemmatized_text = " ".join(lemmatized_sentence)

return lemmatized_text

```

```

[ ]: originaldata["text"] = originaldata.text.apply(lambda x:
    ↪preprocessing_stage2(x))
final["text"] = final.text.apply(lambda x: preprocessing_stage2(x))
testdata["text"] = testdata.text.apply(lambda x: preprocessing_stage2(x))

```

```

[ ]: originaldata

```

```

[ ]:

```

	text	label
0	we remind ourself love mean be will give until...	1
1	most island have differ definit of time access...	0
2	life precious so be babi mother father pleas s...	1
3	too mani peopl be take serious semst	0
4	dude i win freeshirt from i never win anyth lo...	0
..	...	...
482	oh look so not onli be antichoic strong agains...	2
483	i dont think i have right use someon el bodi w...	2
484	peopl arent prolif theyr probirth when mother ...	2
485	i dont follow news there new law all gay peopl...	0
486	god law truth be supress they thing they be et...	0

[487 rows x 2 columns]

```

[ ]: final

```

```

[ ]:

```

	text	label
0	we remind ourself love mean be will give until...	1
1	most island have differ definit of time access...	0
2	life precious so be babi mother father pleas s...	1
3	too mani peopl be take serious semst	0
4	dude i win freeshirt from i never win anyth lo...	0
..	...	...
651	complic come with all type of medic procedur d...	2
652	what doe not reduc abort make illeg what make ...	2
653	so look at not onli antichoic strong against p...	2
654	i dont think i have right use someon el bodi w...	2
655	peopl be not prolifer they be prolifer when mo...	2

[656 rows x 2 columns]

## 1.6 Perform Classification

### 1.7 Original dataset

```
[ ]: def to_list(dataset):  
    labels = dataset['label'].tolist()  
    texts = dataset['text'].tolist()  
  
    print(pd.Series(labels).value_counts())  
  
    return labels, texts
```

```
[ ]: label_list, texts = to_list(originaldata)
```

```
1    219  
0    159  
2    109  
dtype: int64
```

```
[ ]: # Load the BERT tokenizer  
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',  
    ↪do_lower_case=True)
```

```
Downloading (...)solve/main/vocab.txt: 0%|          | 0.00/232k [00:00<?, ?B/s]  
Downloading (...)okenizer_config.json: 0%|          | 0.00/28.0 [00:00<?, ?B/s]  
Downloading (...)lve/main/config.json: 0%|          | 0.00/570 [00:00<?, ?B/s]
```

```
[ ]: max_len = 0  
  
    # For every tweet.  
    for sent in texts:  
  
        # Tokenize the text and add `[CLS]` and `[SEP]` tokens.  
        input_ids = tokenizer.encode(sent, add_special_tokens=True)  
  
        # Update the maximum sentence length.  
        max_len = max(max_len, len(input_ids))  
  
    print('Max sentence length: ', max_len)
```

```
Max sentence length: 41
```

```
[ ]: input_ids = []  
    attention_masks = []
```

```

# For every tweet...
for text in texts:
    encoded_dict = tokenizer.encode_plus(
        text,                                # Sentence to encode.
        add_special_tokens = True, # Add '[CLS]' and '[SEP]'
        max_length = max_len,                # Pad & truncate all
        ↪ sentences.

        truncation = True,
        pad_to_max_length = True,
        return_attention_mask = True, # Construct attn. masks.
        return_tensors = 'pt',         # Return pytorch tensors.
    )

    # Add the encoded sentence to the list.
    input_ids.append(encoded_dict['input_ids'])

    # And its attention mask (simply differentiates padding from non-padding).
    attention_masks.append(encoded_dict['attention_mask'])

# Convert the lists into tensors.
input_ids = torch.cat(input_ids, dim=0)
attention_masks = torch.cat(attention_masks, dim=0)
labels = torch.tensor(label_list)

# Print sentence 0, now as a list of IDs.
print('Original: ', texts[0])
print('Token IDs:', input_ids[0])

```

```

/usr/local/lib/python3.9/dist-
packages/transformers/tokenization_utils_base.py:2354: FutureWarning: The
`pad_to_max_length` argument is deprecated and will be removed in a future
version, use `padding=True` or `padding='longest'` to pad to the longest
sequence in the batch, or use `padding='max_length'` to pad to a max length. In
this case, you can give a specific length with `max_length` (e.g.
`max_length=45`) or leave max_length to None to pad to the maximal input size of
the model (e.g. 512 for Bert).

```

```
warnings.warn(
```

Original: we remind ourself love mean be will give until hurt mother teresa

Token IDs: tensor([ 101, 2057, 10825, 14635, 2884, 2615, 2293, 2812,

2022, 2097,

```

    2507, 2127, 3480, 2388, 12409, 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])

```

```
[ ]: # Combine the training inputs into a TensorDataset.
dataset = TensorDataset(input_ids, attention_masks, labels)

# Create a 80-20 train-validation split.

train_size = int(0.8 * len(dataset))
val_size = len(dataset) - train_size

# Divide the dataset by randomly selecting samples.
train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

print('{:>5,} training samples'.format(train_size))
print('{:>5,} validation samples'.format(val_size))
```

```
389 training samples
 98 validation samples
```

```
[ ]: batch_size = 16
```

```
[ ]: train_dataloader = DataLoader(
    train_dataset, # The training samples.
    sampler = RandomSampler(train_dataset), # Select batches randomly
    batch_size = batch_size # Trains with this batch size.
)

# For validation the order doesn't matter, so we'll just read them sequentially.
validation_dataloader = DataLoader(
    val_dataset, # The validation samples.
    sampler = SequentialSampler(val_dataset), # Pull out batches
    ↪sequentially.
    batch_size = batch_size # Evaluate with this batch size.
)
```

```
[ ]: # Load BertForSequenceClassification, the pretrained BERT model with a single
# linear classification layer on top.

model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased", # Use the 12-layer BERT model, with an uncased vocab.
    num_labels = 3, # The number of output labels--3 for stance classification.
    output_attentions = False, # Whether the model returns attentions weights.
    output_hidden_states = False, # Whether the model returns all hidden-states.
)

model = model.to(device)
```

```
Downloading pytorch_model.bin: 0%|          | 0.00/440M [00:00<?, ?B/s]
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when
```

initializing BertForSequenceClassification:

```
['cls.predictions.transform.LayerNorm.bias', 'cls.predictions.bias',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.weight',
'cls.seq_relationship.bias', 'cls.predictions.transform.LayerNorm.weight']
```

- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized:

```
['classifier.bias', 'classifier.weight']
```

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```
[ ]: optimizer = AdamW(model.parameters(),
                        lr = 2e-5, # args.learning_rate - default is 5e-5, our
                        ↪notebook had 2e-5
                        eps = 1e-8 # args.adam_epsilon - default is 1e-8.
                        )
```

/usr/local/lib/python3.9/dist-packages/transformers/optimization.py:391:  
FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no\_deprecation\_warning=True` to disable this warning

```
warnings.warn(
```

```
[ ]: epochs = 10

# Total number of training steps is [number of batches] x [number of epochs].
total_steps = len(train_dataloader) * epochs

# Create the learning rate scheduler.
scheduler = get_linear_schedule_with_warmup(optimizer,
                                             num_warmup_steps = 0,
                                             num_training_steps = total_steps)
```

```
[ ]: total_steps
```

```
[ ]: 250
```

```
[ ]: # Function to calculate the accuracy of our predictions vs labels
def flat_accuracy(preds, labels):
```

```

pred_flat = np.argmax(preds, axis=1).flatten()
labels_flat = labels.flatten()
return np.sum(pred_flat == labels_flat) / len(labels_flat)

```

```

[ ]: def format_time(elapsed):
    '''
    Takes a time in seconds and returns a string hh:mm:ss
    '''
    # Round to the nearest second.
    elapsed_rounded = int(round((elapsed)))
    # Format as hh:mm:ss
    return str(datetime.timedelta(seconds=elapsed_rounded))

```

```

[ ]: seed_val = 42
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)
training_stats = []
t_acc=[]
t_loss=[]
v_acc=[]
v_loss=[]

# Measure the total training time for the whole run.
total_t0 = time.time()

# For each epoch...
for epoch_i in range(0, epochs):

    #                               Training
    print("")
    print('=====Epoch {:} / {:} ====='.format(epoch_i + 1, epochs))
    print('Training...')

    # Measure how long the training epoch takes.
    t0 = time.time()
    total_train_loss = 0
    total_train_accuracy = 0
    model.train()

    for step, batch in enumerate(train_dataloader):
        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
        optimizer.zero_grad()
        output = model(b_input_ids,

```

```

        token_type_ids=None,
        attention_mask=b_input_mask,
        labels=b_labels)

# ----- Added by me
# Move logits and labels to CPU if we are using GPU
logits = output.logits
logits = logits.detach().cpu().numpy()
label_ids = b_labels.to('cpu').numpy()

total_train_accuracy += flat_accuracy(logits, label_ids)
# -----

loss = output.loss
total_train_loss += loss.item()
# Perform a backward pass to calculate the gradients.
loss.backward()
torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
optimizer.step()
# Update the learning rate.
scheduler.step()

# Report the final accuracy for this validation run.
avg_train_accuracy = total_train_accuracy / len(train_dataloader)
print("  Train Accuracy: {0:.2f}".format(avg_train_accuracy))
# -----

# Calculate the average loss over all of the batches.
avg_train_loss = total_train_loss / len(train_dataloader)

# Measure how long this epoch took.
training_time = format_time(time.time() - t0)
print("")
print("  Average training loss: {0:.2f}".format(avg_train_loss))
print("  Training epoch took: {:}".format(training_time))

#
# Validation
# After the completion of each training epoch, measure our performance on
# the validation set.
print("")
print("Running Validation...")
t0 = time.time()

model.eval()
# Tracking variables

```

```

total_eval_accuracy = 0
best_eval_accuracy = 0
total_eval_loss = 0
nb_eval_steps = 0

# Evaluate data for one epoch
for batch in validation_dataloader:
    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)

    with torch.no_grad():
        output= model(b_input_ids,
                       token_type_ids=None,
                       attention_mask=b_input_mask,
                       labels=b_labels)

        loss = output.loss
        total_eval_loss += loss.item()
        # Move logits and labels to CPU if we are using GPU
        logits = output.logits
        logits = logits.detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()

        total_eval_accuracy += flat_accuracy(logits, label_ids)

# Report the final accuracy for this validation run.
avg_val_accuracy = total_eval_accuracy / len(validation_dataloader)
print(" Accuracy: {0:.2f}".format(avg_val_accuracy))
# Calculate the average loss over all of the batches.
avg_val_loss = total_eval_loss / len(validation_dataloader)
# Measure how long the validation run took.
validation_time = format_time(time.time() - t0)
if avg_val_accuracy > best_eval_accuracy:
    torch.save(model, 'bert_model1')
    best_eval_accuracy = avg_val_accuracy
print(" Validation Loss: {0:.2f}".format(avg_val_loss))

# -----
t_acc.append(avg_train_accuracy)
t_loss.append(avg_train_loss)
v_acc.append(avg_val_accuracy)
v_loss.append(avg_val_loss)
#-----

training_stats.append(

```



```

        {
            'epoch': epoch_i + 1,
            'Training Loss': avg_train_loss,
            'Training Accur': avg_train_accuracy,
            'Valid. Loss': avg_val_loss,
            'Valid. Accur.': avg_val_accuracy,
            'Training Time': training_time,
            'Validation Time': validation_time
        }
    )
    print("")
    print("Training complete!")

    print("Total training took {:} (h:mm:ss)".format(format_time(time.
    ↪time()-total_t0)))

```

===== Epoch 1 / 10 =====

Training...

Train Accuracy: 0.44

Average training loss: 1.06

Training epoch took: 0:00:06

Running Validation...

Accuracy: 0.59

Validation Loss: 0.94

===== Epoch 2 / 10 =====

Training...

Train Accuracy: 0.64

Average training loss: 0.91

Training epoch took: 0:00:03

Running Validation...

Accuracy: 0.61

Validation Loss: 0.88

===== Epoch 3 / 10 =====

Training...

Train Accuracy: 0.72

Average training loss: 0.73

Training epoch took: 0:00:03

Running Validation...

```

Accuracy: 0.48
Validation Loss: 1.12

===== Epoch 4 / 10 =====
Training...
  Train Accuracy: 0.78

  Average training loss: 0.61
  Training epoch took: 0:00:03

Running Validation...
  Accuracy: 0.61
  Validation Loss: 0.88

===== Epoch 5 / 10 =====
Training...
  Train Accuracy: 0.86

  Average training loss: 0.44
  Training epoch took: 0:00:03

Running Validation...
  Accuracy: 0.68
  Validation Loss: 0.81

===== Epoch 6 / 10 =====
Training...
  Train Accuracy: 0.91

  Average training loss: 0.31
  Training epoch took: 0:00:03

Running Validation...
  Accuracy: 0.65
  Validation Loss: 0.88

===== Epoch 7 / 10 =====
Training...
  Train Accuracy: 0.97

  Average training loss: 0.19
  Training epoch took: 0:00:03

Running Validation...
  Accuracy: 0.64
  Validation Loss: 0.92

===== Epoch 8 / 10 =====

```

```

Training..
  Train Accuracy: 0.97

  Average training loss: 0.15
  Training epoch took: 0:00:04

Running Validation...
  Accuracy: 0.65
  Validation Loss: 0.96

===== Epoch 9 / 10 =====
Training..
  Train Accuracy: 0.98

  Average training loss: 0.11
  Training epoch took: 0:00:03

Running Validation...
  Accuracy: 0.59
  Validation Loss: 1.03

===== Epoch 10 / 10 =====
Training..
  Train Accuracy: 0.99

  Average training loss: 0.09
  Training epoch took: 0:00:03

Running Validation...
  Accuracy: 0.62
  Validation Loss: 1.04

Training complete!
Total training took 0:00:54 (h:mm:ss)

```

```
[ ]: from sklearn.metrics import classification_report, confusion_matrix
     from sklearn import metrics
     import matplotlib.pyplot as plt
     import numpy as np
```

```
[ ]: x = np.array([1, 2, 3, 4,5,6,7,8,9,10])
     y = t_acc
     y2 = v_acc

     fig, ax = plt.subplots(figsize=(8,6))

     plt.plot(x, y, label='training_acc')
```

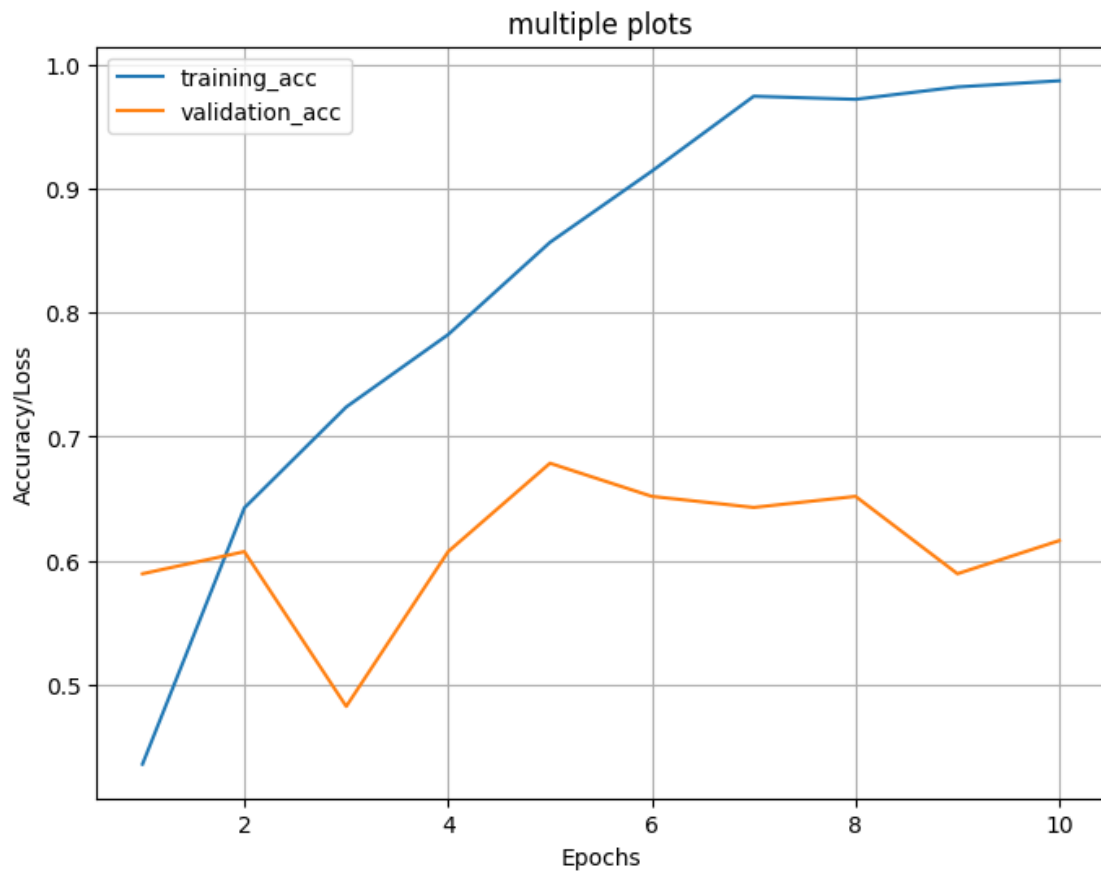
```

plt.plot(x, y2, label='validation_acc')
plt.legend(loc="upper left")
plt.grid()

# for index in range(len(x)):
#     ax.text(x[index], y[index], y[index], size=12)

plt.xlabel("Epochs")
plt.ylabel("Accuracy/Loss")
plt.title('multiple plots')
plt.show()

```



Testing the performance

```
[ ]: testlabels, testtexts = to_list(testdata)
```

```

0    63
1    60
2    58
dtype: int64

```

```
[ ]: test_input_ids = []
test_attention_masks = []
for tweet in testtexts:
    encoded_dict = tokenizer.encode_plus(
        tweet,
        add_special_tokens = True,
        max_length = max_len,
        pad_to_max_length = True,
        return_attention_mask = True,
        return_tensors = 'pt',
    )
    test_input_ids.append(encoded_dict['input_ids'])
    test_attention_masks.append(encoded_dict['attention_mask'])

test_input_ids = torch.cat(test_input_ids, dim=0)
test_attention_masks = torch.cat(test_attention_masks, dim=0)
```

Truncation was not explicitly activated but `max\_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest\_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`.

/usr/local/lib/python3.9/dist-packages/transformers/tokenization\_utils\_base.py:2354: FutureWarning: The `pad\_to\_max\_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max\_length'` to pad to a max length. In this case, you can give a specific length with `max\_length` (e.g. `max\_length=45`) or leave max\_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).

warnings.warn(

```
[ ]: test_dataset = TensorDataset(test_input_ids, test_attention_masks)
test_dataloader = DataLoader(
    test_dataset, # The validation samples.
    sampler = SequentialSampler(test_dataset), # Pull out batches
    ↪sequentially.
    batch_size = batch_size # Evaluate with this batch size.
)
```

```
[ ]: predictions = []
my_predictions=[]
for batch in test_dataloader:
    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    # b_labels = batch[2].to(device)
    with torch.no_grad():
```

```

output= model(b_input_ids,
               token_type_ids=None,
               attention_mask=b_input_mask)

logits = output.logits
logits = logits.detach().cpu().numpy()
pred_flat = np.argmax(logits, axis=1).flatten()

predictions.extend(list(pred_flat))

```

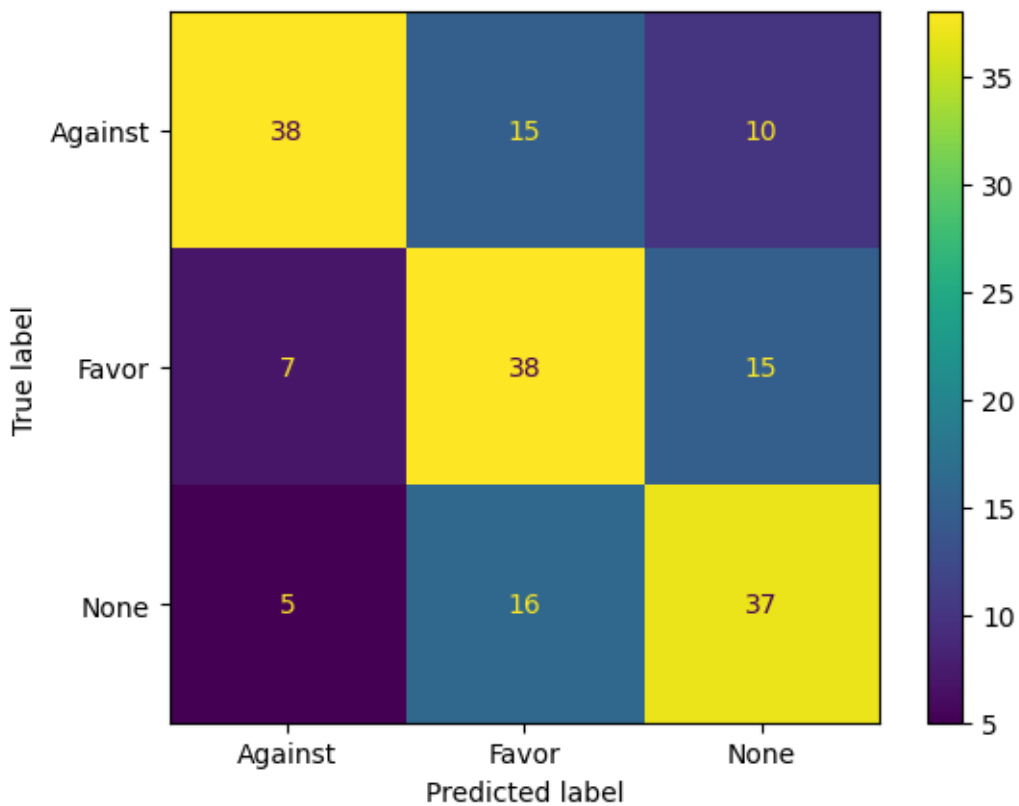
```

[ ]: target_names = ['Against', 'Favor', 'None']

confusion_matrix = metrics.confusion_matrix(testlabels, predictions)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
    ↳confusion_matrix, display_labels=target_names)
cm_display.plot()
plt.show()

print(classification_report(testlabels, predictions, target_names=target_names))

```



```

precision    recall  f1-score   support

```

Against	0.76	0.60	0.67	63
Favor	0.55	0.63	0.59	60
None	0.60	0.64	0.62	58
accuracy			0.62	181
macro avg	0.64	0.62	0.63	181
weighted avg	0.64	0.62	0.63	181

## 1.8 Classification after augmentation

```
[ ]:
```

```
[ ]: label_list, texts = to_list(final)
```

```
1    219
0    219
2    218
dtype: int64
```

```
[ ]: max_len = 0
```

```
for sent in texts:

    input_ids = tokenizer.encode(sent, add_special_tokens=True)

    max_len = max(max_len, len(input_ids))

print('Max sentence length: ', max_len)
```

```
Max sentence length: 41
```

```
[ ]: input_ids = []
```

```
attention_masks = []
```

```
for text in texts:
    encoded_dict = tokenizer.encode_plus(
        text,
        add_special_tokens = True,
        max_length = max_len,
        truncation = True,
        pad_to_max_length = True,
        return_attention_mask = True,
        return_tensors = 'pt',
    )
```

```

input_ids.append(encoded_dict['input_ids'])

attention_masks.append(encoded_dict['attention_mask'])

input_ids = torch.cat(input_ids, dim=0)
attention_masks = torch.cat(attention_masks, dim=0)
labels = torch.tensor(label_list)

print('Original: ', texts[0])
print('Token IDs:', input_ids[0])

```

/usr/local/lib/python3.9/dist-packages/transformers/tokenization\_utils\_base.py:2354: FutureWarning: The `pad\_to\_max\_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max\_length'` to pad to a max length. In this case, you can give a specific length with `max\_length` (e.g. `max\_length=45`) or leave max\_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).

```
warnings.warn(
```

```

Original:  we remind ourself love mean be will give until hurt mother teresa
Token IDs: tensor([ 101,  2057, 10825, 14635,  2884,  2615,  2293,  2812,
 2022,  2097,
        2507,  2127,  3480,  2388, 12409,   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])

```

```

[ ]: dataset = TensorDataset(input_ids, attention_masks, labels)

train_size = int(0.8 * len(dataset))
#val_size = int(0.2 * len(dataset))
val_size = len(dataset) - train_size

train_dataset, val_dataset = random_split(dataset, [train_size, val_size])

print('{:>5,} training samples'.format(train_size))
print('{:>5,} validation samples'.format(val_size))

```

```

524 training samples
132 validation samples

```

```

[ ]: train_dataloader = DataLoader(
    train_dataset,

```



```

        sampler = RandomSampler(train_dataset),
        batch_size = batch_size
    )

validation_dataloader = DataLoader(
    val_dataset,
    sampler = SequentialSampler(val_dataset),
    batch_size = batch_size

```

```

[ ]: model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels = 3,
    output_attentions = False,
    output_hidden_states = False,
)

model = model.to(device)

```

Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForSequenceClassification:

```

['cls.predictions.transform.LayerNorm.bias', 'cls.predictions.bias',
'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.weight',
'cls.predictions.transform.dense.bias', 'cls.seq_relationship.weight',
'cls.seq_relationship.bias', 'cls.predictions.transform.LayerNorm.weight']

```

- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).

- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized:

```

['classifier.bias', 'classifier.weight']

```

You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

```

[ ]: total_steps = len(train_dataloader) * epochs

scheduler = get_linear_schedule_with_warmup(optimizer,
                                             num_warmup_steps = 0, # Default
                                             ↪value in run_glue.py
                                             num_training_steps = total_steps)

```

```

[ ]: total_steps

```

[ ]: 330

```
[ ]: # Function to calculate the accuracy of our predictions vs labels
def flat_accuracy(preds, labels):
    pred_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return np.sum(pred_flat == labels_flat) / len(labels_flat)
```

```
[ ]: def format_time(elapsed):
    '''
    Takes a time in seconds and returns a string hh:mm:ss
    '''
    # Round to the nearest second.
    elapsed_rounded = int(round((elapsed)))
    # Format as hh:mm:ss
    return str(datetime.timedelta(seconds=elapsed_rounded))
```

```
[ ]: seed_val = 42
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)
training_stats = []
t_acc=[]
t_loss=[]
v_acc=[]
v_loss=[]

total_t0 = time.time()

# For each epoch...
for epoch_i in range(0, epochs):

    # Training

    print("")
    print('=====Epoch {:} / {:} ====='.format(epoch_i + 1, epochs))
    print('Training...')

    t0 = time.time()
    total_train_loss = 0
    total_train_accuracy = 0
    model.train()
```

```

for step, batch in enumerate(train_dataloader):

    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)
    optimizer.zero_grad()
    output = model(b_input_ids,
                    token_type_ids=None,
                    attention_mask=b_input_mask,
                    labels=b_labels)

    logits = output.logits
    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()

    total_train_accuracy += flat_accuracy(logits, label_ids)
    # -----

    loss = output.loss
    total_train_loss += loss.item()

    loss.backward()

    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

    optimizer.step()

    scheduler.step()

avg_train_accuracy = total_train_accuracy / len(train_dataloader)
print("  Train Accuracy: {0:.2f}".format(avg_train_accuracy))

avg_train_loss = total_train_loss / len(train_dataloader)

training_time = format_time(time.time() - t0)
print("")
print("  Average training loss: {0:.2f}".format(avg_train_loss))
print("  Training epoch took: {:}".format(training_time))

print("")
print("Running Validation...")
t0 = time.time()

```

```

model.eval()
# Tracking variables
total_eval_accuracy = 0
best_eval_accuracy = 0
total_eval_loss = 0
nb_eval_steps = 0

# Evaluate data for one epoch
for batch in validation_dataloader:
    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)

    with torch.no_grad():
        output= model(b_input_ids,
                       token_type_ids=None,
                       attention_mask=b_input_mask,
                       labels=b_labels)

        loss = output.loss
        total_eval_loss += loss.item()

        logits = output.logits
        logits = logits.detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()

        total_eval_accuracy += flat_accuracy(logits, label_ids)

avg_val_accuracy = total_eval_accuracy / len(validation_dataloader)
print(" Accuracy: {0:.2f}".format(avg_val_accuracy))

avg_val_loss = total_eval_loss / len(validation_dataloader)

validation_time = format_time(time.time() - t0)
if avg_val_accuracy > best_eval_accuracy:
    torch.save(model, 'bert_model1')
    best_eval_accuracy = avg_val_accuracy
print(" Validation Loss: {0:.2f}".format(avg_val_loss))

# -----
t_acc.append(avg_train_accuracy)
t_loss.append(avg_train_loss)
v_acc.append(avg_val_accuracy)
v_loss.append(avg_val_loss)
#-----

```

```

training_stats.append(
    {
        'epoch': epoch_i + 1,
        'Training Loss': avg_train_loss,
        'Training Accur': avg_train_accuracy,
        'Valid. Loss': avg_val_loss,
        'Valid. Accur.': avg_val_accuracy,
        'Training Time': training_time,
        'Validation Time': validation_time
    }
)
print("")
print("Training complete!")

print("Total training took {:} (h:mm:ss)".format(format_time(time.
↪time()-total_t0)))

```

===== Epoch 1 / 10 =====

Training...

Train Accuracy: 0.31

Average training loss: 1.19

Training epoch took: 0:00:04

Running Validation...

Accuracy: 0.36

Validation Loss: 1.17

===== Epoch 2 / 10 =====

Training...

Train Accuracy: 0.32

Average training loss: 1.18

Training epoch took: 0:00:04

Running Validation...

Accuracy: 0.36

Validation Loss: 1.17

===== Epoch 3 / 10 =====

Training...

Train Accuracy: 0.32

Average training loss: 1.18

Training epoch took: 0:00:04

```

Running Validation...
  Accuracy: 0.36
  Validation Loss: 1.17

===== Epoch 4 / 10 =====
Training...
  Train Accuracy: 0.32

  Average training loss: 1.18
  Training epoch took: 0:00:04

Running Validation...
  Accuracy: 0.36
  Validation Loss: 1.17

===== Epoch 5 / 10 =====
Training...
  Train Accuracy: 0.33

  Average training loss: 1.18
  Training epoch took: 0:00:05

Running Validation...
  Accuracy: 0.36
  Validation Loss: 1.17

===== Epoch 6 / 10 =====
Training...
  Train Accuracy: 0.33

  Average training loss: 1.19
  Training epoch took: 0:00:04

Running Validation...
  Accuracy: 0.36
  Validation Loss: 1.17

===== Epoch 7 / 10 =====
Training...
  Train Accuracy: 0.32

  Average training loss: 1.19
  Training epoch took: 0:00:04

Running Validation...
  Accuracy: 0.36
  Validation Loss: 1.17

```

===== Epoch 8 / 10 =====

Training...

Train Accuracy: 0.32

Average training loss: 1.19

Training epoch took: 0:00:04

Running Validation...

Accuracy: 0.36

Validation Loss: 1.17

===== Epoch 9 / 10 =====

Training...

Train Accuracy: 0.32

Average training loss: 1.18

Training epoch took: 0:00:04

Running Validation...

Accuracy: 0.36

Validation Loss: 1.17

===== Epoch 10 / 10 =====

Training...

Train Accuracy: 0.32

Average training loss: 1.17

Training epoch took: 0:00:04

Running Validation...

Accuracy: 0.36

Validation Loss: 1.17

Training complete!

Total training took 0:01:01 (h:mm:ss)

```
[ ]: x = np.array([1, 2, 3, 4,5,6,7,8,9,10])
      y = t_acc
      y2 = v_acc

      fig, ax = plt.subplots(figsize=(8,6))

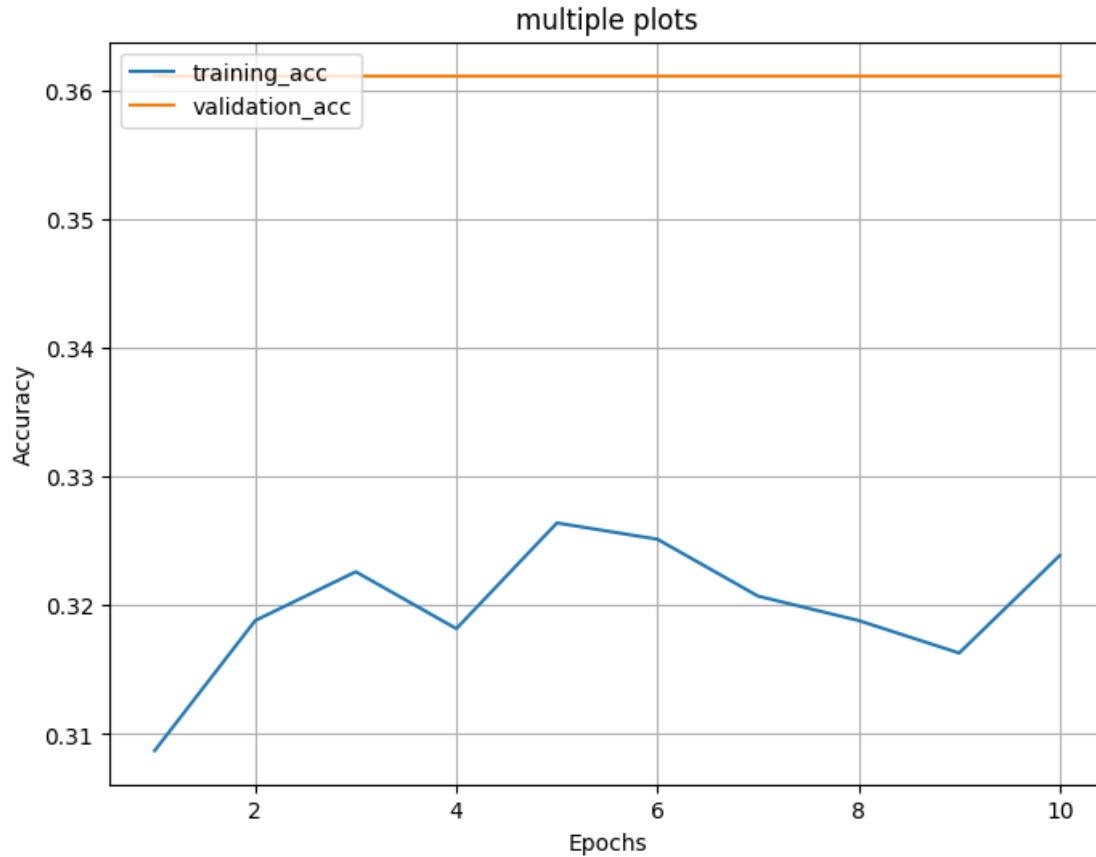
      plt.plot(x, y, label='training_acc')
      plt.plot(x, y2, label='validation_acc')
      plt.legend(loc="upper left")
      plt.grid()
```

```

# for index in range(len(x)):
#     ax.text(x[index], y[index], y[index], size=12)

plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title('multiple plots')
plt.show()

```



Testing the performance

```
[ ]: testlabels, testtexts = to_list(testdata)
```

```

0    63
1    60
2    58
dtype: int64

```

```

[ ]: test_input_ids = []
     test_attention_masks = []

```



```

for tweet in testtexts:
    encoded_dict = tokenizer.encode_plus(
        tweet,
        add_special_tokens = True,
        max_length = max_len,
        pad_to_max_length = True,
        return_attention_mask = True,
        return_tensors = 'pt',
    )
    test_input_ids.append(encoded_dict['input_ids'])
    test_attention_masks.append(encoded_dict['attention_mask'])

test_input_ids = torch.cat(test_input_ids, dim=0)
test_attention_masks = torch.cat(test_attention_masks, dim=0)

```

/usr/local/lib/python3.9/dist-packages/transformers/tokenization\_utils\_base.py:2354: FutureWarning: The `pad\_to\_max\_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max\_length'` to pad to a max length. In this case, you can give a specific length with `max\_length` (e.g. `max\_length=45`) or leave max\_length to None to pad to the maximal input size of the model (e.g. 512 for Bert).

```
warnings.warn(
```

```

[ ]: test_dataset = TensorDataset(test_input_ids, test_attention_masks)
test_dataloader = DataLoader(
    test_dataset, # The validation samples.
    sampler = SequentialSampler(test_dataset), # Pull out batches
    ↪sequentially.
    batch_size = batch_size # Evaluate with this batch size.
)

```

```

[ ]: predictions = []
my_predictions=[]
for batch in test_dataloader:
    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    # b_labels = batch[2].to(device)
    with torch.no_grad():
        output= model(b_input_ids,
                       token_type_ids=None,
                       attention_mask=b_input_mask)
        logits = output.logits
        logits = logits.detach().cpu().numpy()
        pred_flat = np.argmax(logits, axis=1).flatten()

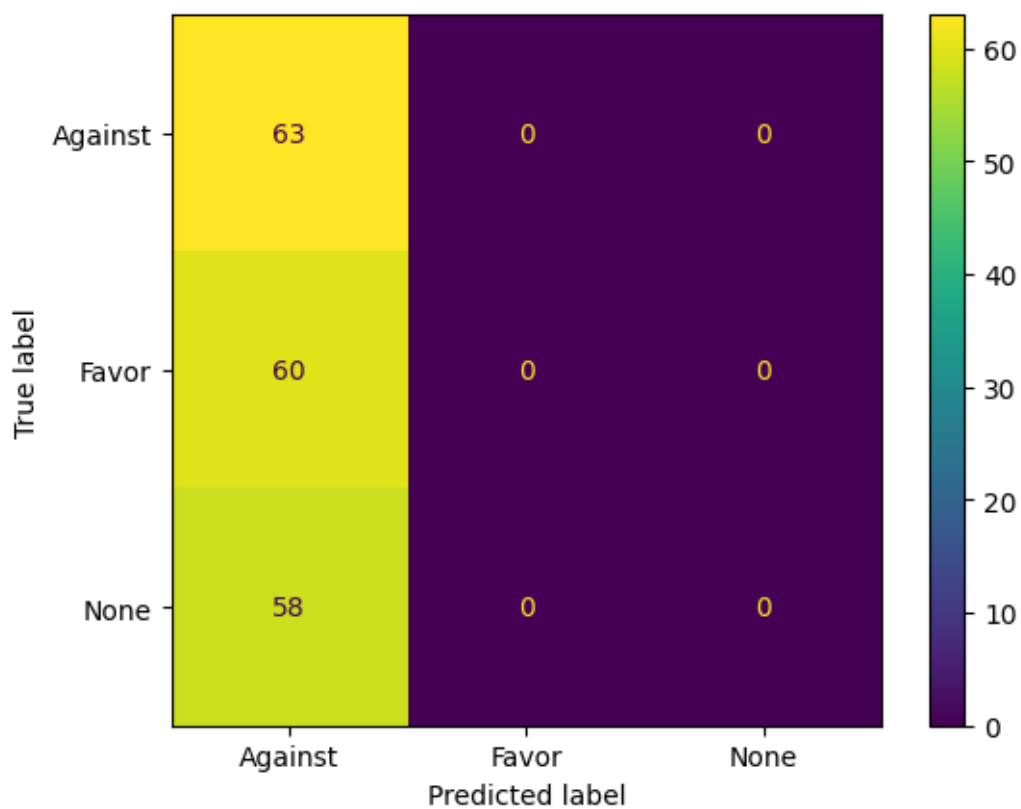
```

```
predictions.extend(list(pred_flat))
```

```
[ ]: target_names = ['Against', 'Favor', 'None']

confusion_matrix = metrics.confusion_matrix(testlabels, predictions)
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
    ↳confusion_matrix, display_labels=target_names)
cm_display.plot()
plt.show()

print(classification_report(testlabels, predictions, target_names=target_names))
```



	precision	recall	f1-score	support
Against	0.35	1.00	0.52	63
Favor	0.00	0.00	0.00	60
None	0.00	0.00	0.00	58
accuracy			0.35	181
macro avg	0.12	0.33	0.17	181

weighted avg            0.12            0.35            0.18            181

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
```

```
    _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
```

```
    _warn_prf(average, modifier, msg_start, len(result))
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
```

```
    _warn_prf(average, modifier, msg_start, len(result))
```