Student: Megh Shah
Mentor: Dr. Tara Sainath, Google, Inc.
https://sites.google.com/site/tsainath/

SAIDL Induction Documentation:

2. Part 2:

Basic Overview:

- We aim to create a model which can classify various sentences to HATE SPEECH/NOT HATE SPEECH.
- For this particular model, we are making use of the HASOC dataset.
- At a high level, we are translating a part of an English sentence into a Hindi(for various CMI values). CMI: The fraction of the sentence that has to be translated to Hindi. Here the CMI takes the following values: 0.2, 0.25, 0.5, 0.75 and 1.00.
- We train it using BERT(Bidirectional Encoder Representations for Transformers).
- We need to convert our dataset into a format on which BERT can be trained on.
- For that purpose we use something known as Tokenizing.
- After we train the model, we move forward to test it.
- We tested the model for **five values** of CMI. We also used two models of BERT.
- The table below lists the efficiency of the model for each value of CMI for each of the BERT models namely 'bert-base-multilingual-cased' and ''.

Explanation of the code:

- In the very first part of the code, we install google translate(install googletrans) and transformer.
- I also mounted(linked) my Google Drive to Colab in order to import some data files required.
- Then we import a set of packages which are necessary for the completion of our task.
- In the next step, we read our original data file (`english_2021.csv`) which has to be translated.
- After this is done, we actually translate the file according to the set CMI and save the file(`english_2021_translated_1p00.csv`).
- We have taken the translation of the sentence in the **Latin** script instead of the original **Devanagiri** script used in writing Hindi. This is done because the model can only train on Latin script and not on the typical Devanagiri one.
- The word to word translations have been taken for all the sentences. Not all sentences translated make complete sense as the sentence structure has not been considered for simplicity.

For example, lets take the sentence “Good morning how was your day?”
Code-mixing this becomes “Su prabhat how was your din?”

- Now we will use the translated data for training,validation and test purposes.
- The next block of code divides the whole dataset into training data(80%), validation data(10%) and test data(10%).

```
Number of training sentences: 3,843
Training: 3074
Validation: 384
Test: 385
```

- Then the tokenizing takes(explained above) place. Each word is expressed with a unique value which is then multiplied with a Single-Hot Vector.
- After this step, the actual training takes place(in multiple epochs).
- In the final few steps, training and validation plots are plotted.
- Finally, the testing takes place which gives us a resulting test efficiency for each value of CMI. The results are displayed in the table below.

Results:

	CMI	Accuracy(%)
BERT	0.00	79.80
	0.25	77.60
	0.50	76.40
	0.75	66.60
	1.00	66.10
mBERT	0.00	75.70
	0.25	76.00
	0.50	74.00
	0.75	64.90
	1.00	65.60
XLM-17	0.00	77.60
	0.25	78.60
	0.50	78.40
	0.75	66.80
	1.00	64.40

Trend of the Result:

- For both the cases mentioned above, we see that the efficiency increases initially with the CMI, reaches a maxima and then shows a steady decline as the CMI increases further.
- Case 1: The efficiency peaks at 79.80% when the CMI = 0.00 and then goes on decreasing to 66.10% as the CMI tends to 1.00.
  We see a visible drop in accuracy as the CMI goes up because it is harder for the model to work/predict on Hindi words as the model is mainly trained in the language English.
  Hence we see a decrease in the accuracy as the CMI increases i.e. number of Hindi words in the sentence increases.
- Case 2: Similar trend is visible for the second case as well. The efficiency slightly increases to 76.00% at CMI = 0.25 and then subsides to 65.60% at CMI = 1.00.
  We notice the same trend in mBERT as well because of the same reason as mentioned in the previous case i.e. the model has been mainly/predominantly been trained in English so it is harder for it to predict when the Hindi words in a particular sentence increases(CMI index increases).
- Case 3: Similar trend is visible for the third case as well. The efficiency slightly increases to 78.60% at CMI = 0.25 and then subsides to 64.40% at CMI = 1.00.
  We notice the same trend in this model as well because of the same reason as mentioned in the previous case i.e. the model has been mainly/predominantly been trained in English so it is harder for it to predict when the Hindi words in a particular sentence increases(CMI index increases). This one uses the XLM-17 model which is trained in 17 languages.
- The BERT model is trained in only ENGLISH Language and so the accuracy is highest when the CMI = 0.
- Now when we have the value of CMI increasing, the accuracy of mBERT and XLM-17 is higher as those models are trained in multilingual languages.
- Now, when the CMI is equal to 1, the performance of all the three models is relatively unsatisfactory/low.

- The common features of all the models are:
  1) Efficiency is highest at CMI = 0.25 or CMI = 0.00 in all cases.
  2) CMI = 1.00 is the point where the accuracy is the lowest.
- Bert is slightly better than mBert. This might be because BERT is trained in Latin Script while mBERT is trained in Devanagari Script and our sentence is wholly in the Latin Script.

References:

- https://www.kaggle.com/code/jaskaransingh/bert-fine-tuning-with-pytorch
- https://github.com/Kalit31/HASOC-2021/blob/main/English/BERT%20based/finetuned-bert.ipynb
- https://vaclavkosar.com/ml/Tokenization-in-Machine-Learning-Explained#:~:text=Tokenization%20is%20splitting%20the%20input,embedded)%20into%20a%20vector%20space.
- https://towardsdatascience.com/how-to-build-a-wordpiece-tokenizer-for-bert-f505d97dddbb
- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270