Pibit.ai - Assignment: Cause & Body Part Classification
By: Tanishq Selot, IIT Indore
Colab Notebook (Framework - Pytorch) - here

EDA:

The exploratory data analysis started with converting the string valued rows to lower case as many of the classes were similar but differed by just letter casing. Moreover the classes also differed by ‘commas’ as well. Lowercasing brought down the number of causes from 16 to 11 and the number of body part classes from 8 to 7. They are as follows:

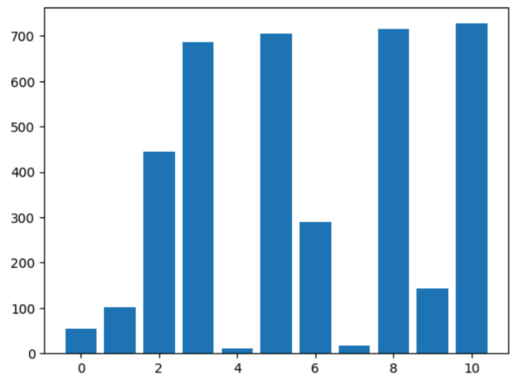
Causes:

1. 'burn or scald - heat or cold exposures - contact with'
2. 'caught in under or between'
3. 'cut puncture scrape injured by'
4. 'fall slip or trip injury'
5. 'includes freezing'
6. 'misc'
7. 'motor vehicle'
8. 'rubbed or abraded by'
9. 'strain or injury by'
10. 'striking against or stepping on'
11. 'struck or injured by'

Body Parts:

1. 'head'
2. 'lower extremities'
3. 'misc'
4. 'multiple body parts'
5. 'neck'
6. 'trunk'
7. 'upper extremities'

A total of 3918 data points are given for training. Upon eliminating the NaN labels, 3892 data points were left for cause classification and 3659 were left for Body Part classification. A clear class imbalance can also be seen in case of cause classification (11 classes):



Due to time constraints and clash with my academic work, I was not able to try resampling and weighted loss function methods to tackle the issue of class imbalance. Finally the columns “LossDescription”, “PartInjured” and “ResultingInjuries” were combined into one sentence after the following preprocessing steps:

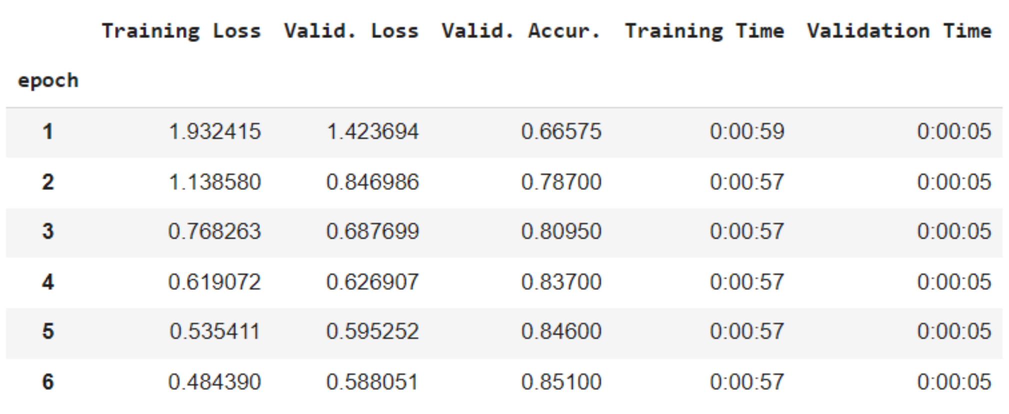
1. Cleaning Numbers
2. Cleaning alpha-numeric characters
3. Cleaning repeating characters

The labels were one-hot encoded.

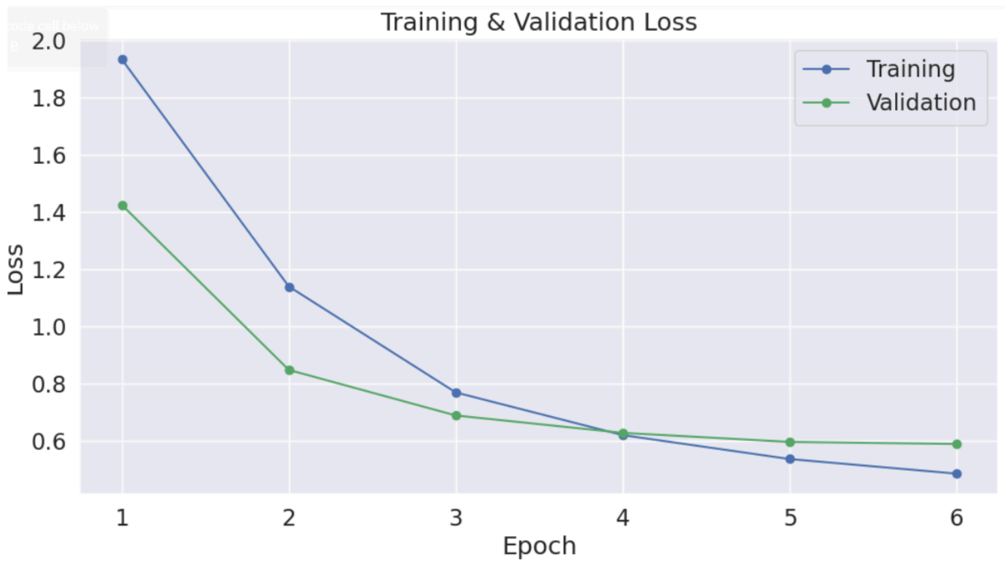
Approach 1:

Treating the problem like a sentiment analysis case (classification of the combined text), BERT (Bidirectional Encoder Representation of Transformer) was fine-tuned for 6 epochs and the following results were obtained based on a 80-20 training-validation split:

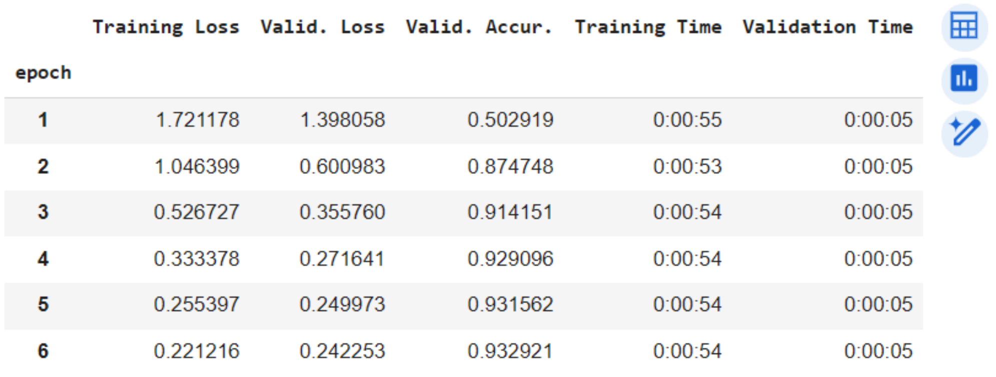
Cause Classification (Could have trained for more epochs but GPU limitations were encountered with Colab):



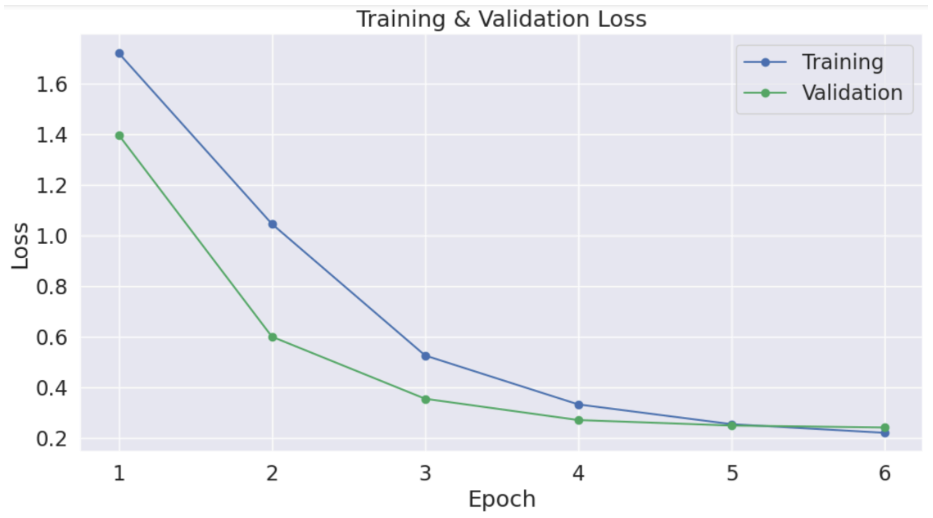
epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	1.932415	1.423694	0.66575	0:00:59	0:00:05
2	1.138580	0.846986	0.78700	0:00:57	0:00:05
3	0.768263	0.687699	0.80950	0:00:57	0:00:05
4	0.619072	0.626907	0.83700	0:00:57	0:00:05
5	0.535411	0.595252	0.84600	0:00:57	0:00:05
6	0.484390	0.588051	0.85100	0:00:57	0:00:05



Body Part Classification:



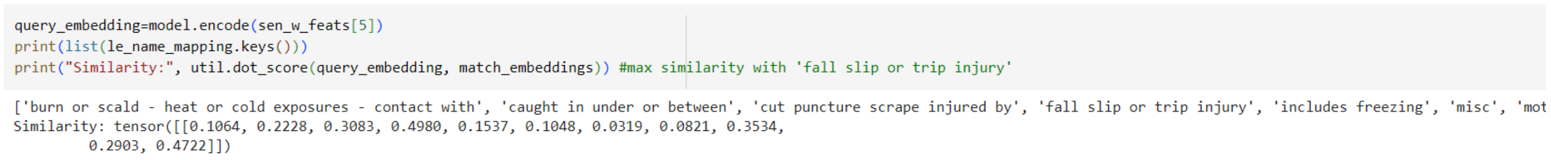
epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	1.721178	1.398058	0.502919	0:00:55	0:00:05
2	1.046399	0.600983	0.874748	0:00:53	0:00:05
3	0.526727	0.355760	0.914151	0:00:54	0:00:05
4	0.333378	0.271641	0.929096	0:00:54	0:00:05
5	0.255397	0.249973	0.931562	0:00:54	0:00:05
6	0.221216	0.242253	0.932921	0:00:54	0:00:05



The final test file for this approach can be found at bert_results.csv. Better results than approach 2 were obtained.

Approach 2:

Treating the problem as a semantic similarity case due to high correlation between “PartInjured” columns and “Body Part - Hierarchy 1” columns and also between the other two independent columns and “Cause - Hierarchy 1”, `multi-qa-MiniLM-L6-cos-v1 (not trained on the data)` was used to find the similarity scores of the combined text with all the class texts.



```
query_embedding=model.encode(sen_w_feats[5])
print(list(le_name_mapping.keys()))
print("Similarity:", util.dot_score(query_embedding, match_embeddings)) #max similarity with 'fall slip or trip injury'
```

```
['burn or scald - heat or cold exposures - contact with', 'caught in under or between', 'cut puncture scrape injured by', 'fall slip or trip injury', 'includes freezing', 'misc', 'mot
Similarity: tensor([[0.1064, 0.2228, 0.3083, 0.4980, 0.1537, 0.1048, 0.0319, 0.0821, 0.3534,
         0.2903, 0.4722]])
```

The maximum score class was then selected. I was also not able to train this model due to time constraints. The final test file for this approach can be found at semantic_results_untrained.csv.

Future Scope:

1. Tackling class imbalance
2. Training for more epochs
3. Training the semantic similarity model