CS728 Assigment 2 Report

Singamsetty Sandeep (213050064)
Saswat Meher (22m0804)

Q1) All-to-All Attention Using Bert-tiny for STS task.

We used bert-tiny to get the embeddings for tokens in a sentence. Both of the sentence Sent1 and Sent2 are send at a time to the bert-tiny. Then used the pooled output from the bert model which is basically mean of all the output embeddings. On top of the pooled output use a Linear layer get a scalar value to denote the similarity. Also a dropout layer was a added before Linear layer. Used MSE loss between predicted value and labels in training data.

Training Procedure:
First 10 epochs were run with freezing the bert model params so that the new linear layer params are not completely random and somewhat aligned to out requirement. Then for the rest of the epoch bert-model params are made trainable. This way we also finetune the Bert-tiny too.

Validation Score:
{'pearson': 0.7717147655332827, 'spearmanr': 0.7812636986063088}

Inference:
Here is an example of similarity score of 2 sentence pairs.

Sent1: I am playing guitar.
Sent2: He is using a guitar.

```
{'input_ids': tensor([[ 101, 1045, 2572, 2652, 2858, 1012,  102, 2002, 2003, 2478, 1037, 2858,
        1012,  102]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1]]), 'attention_mask':
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])}
```

```
Sim Score: tensor([[3.2934]])
```

Q3) DTW

We used bert-tiny to get the embeddings for tokens in a sentence. Both of the sentence Sent1 and Sent2 are sent to the bert-tiny separately to get the contextual embedings of tokens separately.
Implemented Dynamic Time Warping to get the similarity between the list of vectors of each sentence. Similarity between 2 vectors were calculated with cosine similarity and then a tanh layer on top of it after scaling it with “a” and “b”.

3.1) Non_crossing implementation

Non-crossing was maintained using the Algorithm mentioned on the midsem paper.

Training Procedure:
Freezed the bert model params as we are only intrested in training a,b. Labels were normalised as (x*2)/5 -1 to get every labels in the range of (-1,1) which is the case for tanh.

Validation Score:
Sim Metric: {'pearson': 0.49427789128215127, 'spearmanr': 0.5871720153399248}

Inference:
Here is an example of similarity score and alignment of 2 sentence pairs.
The two sentence used have allignment of word ball and break as non crossing and it gives a good score.

Sent 1: He used a ball to break into the house.
Sent 2: A man used a steel ball to break into the glass house.

```
['a', 'man', 'used', 'a', 'steel', 'ball', 'to', 'break', 'into', 'the', 'glass', 'house', '.']
['he', 'used', 'a', 'ball', 'to', 'break', 'into', 'the', 'house', '.']

Sim Score: tensor(7.5739)

Allignment: tensor([ 0,  2,  3,  5,  6,  7,  8,  9, 11, 12])

he      :   a
used    :   used
a       :   a
ball    :   ball
to      :   to
break   :   break
into    :   into
the     :   the
house   :   house
.       :   .
```

Where if provided a sentence with alligned tokens with crossing in it, it is not good at predicting the allignments. In the below example Break and Ball are crossing, and the model confuses itself while alligning the two words.

Sent1: He used a ball to break into the house.
Sent2: He break into the house using a ball.

```
['he', 'used', 'a', 'ball', 'to', 'break', 'into', 'the', 'house', '.']
['he', 'break', 'into', 'the', 'house', 'using', 'a', 'ball', '.']

Sim Score: tensor(5.8867)
Allignment: [0, 1, 2, 4, 5, 6, 7, 8, 9]

he      :   he
break   :   used
into    :   a
the     :   to
house   :   break
using   :   into
a       :   the
ball    :   house
.       :   .
```

<u>3.2) Crossing implementation</u>

As here we are allowed to have crossing between the alignment of tokens, the similarity score are basically nothing but the sum of cosine similarity of each token in sent1 to the most similar token in sent2.

Training Procedure:

Freezed the bert model params as we are only intrested in training a,b. Labels were normalised as (x*2)/5 -1 to get every labels in the range of (-1,1) which is the case for tanh.

Validation Score:
Sim Metric: {'pearson': 0.6054388173862182, 'spearmanr': 0.6063385252022535}

Inference:

Here is an example of similarity score and alignment of 2 sentence pairs. Allignment is from the smaller sentence to larger sentence.

Sent1: he used a ball to break into the house.
Sent2: He break into the house using a ball.

```
['he', 'used', 'a', 'ball', 'to', 'break', 'into', 'the', 'house', '.']
['he', 'break', 'into', 'the', 'house', 'using', 'a', 'ball', '.']

Sim Score: tensor([3.0479])
Allignment: tensor([0, 5, 6, 7, 8, 6, 2, 3, 9])

he      :   he
break   :   break
into    :   into
the     :   the
house   :   house
using   :   into
a       :   a
ball    :   ball
.       :   .
```

<u>Analysis:</u>

Bert-Tiny with all to all attention between token of both the sentence was able to achieve a very good performance of around .8 correlation coeffient. Also for the first 10 epochs where BERT model params were freezed it was able to achieve a maximum correlation coeff of around 0.7.
Where as the implementation using DTW to calculate the similarity score between two sentence was not able to achieve best of the performance with only 0.5 corr coeff between predicted and actual similarity score. The results are so maybe due to the constraint we are imposing on the allignment of the tokens in two sentences that might be making it hard for the model to get the actual similarity score .
On the other hand where crossings were allowed the model performed as similar to that of Bert-tiny without the finetuning of BERT params i.e. around 0.7 percent.