

CS728 - Organisation Of Web Information
Assignment 2
Spring 2023
Adarsh Raj, 190050004
Tanu Goyal, 190050123

Dynamic Time Warping (DTW) With Limited Crossings

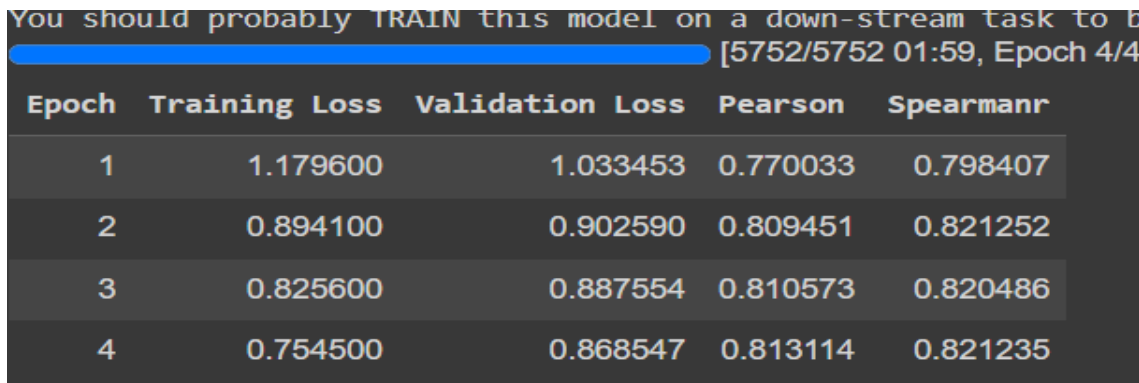
Date: April 15, 2023

Q1: Early Interaction using BERT-tiny

Each dataset sample consists of two sentences and an associated similarity score. We fine BERT-tiny to assign similarity score given a pair of sentences as inputs. To ensure early interaction between the input sentences, we concatenate the two sentences separated by a *[CLS]* token and pass this as an input to the model. The final/output layer of the model is a text classification layer with number of outputs set to 1. This acts as a regression layer. Regression is required for the task since the output space is continuous rather than discrete. We deploy HuggingFace's inbuilt hyper-parameter search methods for hyper-parameter tuning.

We refer to [Text Classification using BERT](#) for implementing this part of the assignment.

We perform inference using *'The man hit the other man with a stick.'* as both sentence 1 and sentence 2. The the score assigned for this is around 2.5 which deviates from the ground truth score of 5.0. This highlights that BERT-tiny fails to learn the task well.



The image shows a terminal window with a blue progress bar at the top indicating 5752/5752 steps completed in 01:59 at Epoch 4/4. Below the progress bar is a table with five columns: Epoch, Training Loss, Validation Loss, Pearson, and Spearmanr. The table contains four rows of data for epochs 1 through 4, showing a decrease in loss and an increase in correlation coefficients over time.

Epoch	Training Loss	Validation Loss	Pearson	Spearmanr
1	1.179600	1.033453	0.770033	0.798407
2	0.894100	0.902590	0.809451	0.821252
3	0.825600	0.887554	0.810573	0.820486
4	0.754500	0.868547	0.813114	0.821235

Figure 1: Loss and correlation coefficient values across tiny-BERT fine-tuning epochs

Q2: DTW With Non Crossing

Implementation Details

We implemented a Dynamic Programming approach to calculate the similarity score and alignment map for this part of the assignment. The Dynamic programming function `dtw`, follows the logic of the solution of midsem linked in the assignment. Alignment is also calculated alongside DTW in the same DP approach.

For this part of the assignment, bunch of functions were defined which are as follows:

- `encode_sentence`: This is used for inputting a single sentence to gets its embeddings vector from model. The result is of size `(len(sentence), 128)` from the `bert-tiny` model.
- `cosine_similarity`: This function is used for calculating the cosine similarity given two embeddings.
- `compute_cost_matrix`: This function is used for calculating the embeddings and returning the similarity matrix used in DTW DP.

DTW: This main function uses the above function and DP approach to calculate the alignment 2D map and similarity scores given two sentences. This function is then also used for test sentences, to calculate their mapping.

Analysis

```
s1 = "This is a test sentence"
s2 = "This sentence is a test"

DTW, align = dtw(s1, s2)
print("DTW Score:", DTW[-1, -1]) # Final Score
print("Alignment Map:", align) # Alignment MAP

## The alignment map contains indices of most possible matching. For each row, i -> j
print("Alignment:", align[:, -1])
```

```
DTW Score: 4.336558163166046
Alignment Map: [[ 0  0  0  0  0  0]
 [ 0  1  1  1  1  1]
 [ 0 -1  2  3  3  3]
 [ 0 -1 -1  3  4  4]
 [ 0 -1 -1 -1  4  5]
 [ 0 -1  2 -1  4  5]]
Alignment: [0 1 3 4 5 5]
```

Figure 2: DTW test for two sentences

Qualitative Analysis: Note that, here we are getting a similarity score of 4.34 and also the alignment map `[0,1,3,4,5,5]`. Ignoring the first and last term, since they corresponds to extra start and end terms which are added to calculate embeddings by model, we get mappings, for `This->1`, `is->3`, `a->4`, `test->5`, which is a correct non-crossing mapping.

Q3: Tanh Calibration in DTW

Implementation Details

We implemented a small modified version of the DTW used in question 2. This calculates the similarity matrix using parameterized tanh calibration. Both the crossing and non-crossing similarity score functions for this part are modified such that all the variables are tensor, not numpy, to be supported by backward propagation while training. We don't need alignment for this part, so it is omitted from the code and a normalised similarity score is being outputted by the DP functions.

Normalisation: It is required to make scores compatible and uniform with the labels already given. The normalization done is $\text{DTW}(\mathbf{I}, \mathbf{J}) / \min(\mathbf{I}, \mathbf{J})$, that is we are dividing the score by the length of smaller sentence.

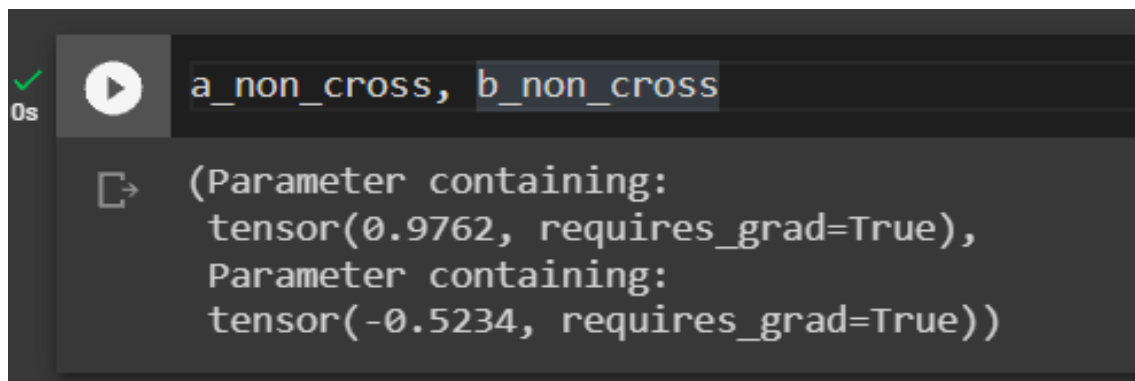
Q3.1: Non Crossing

We used DP (function name: `dtw_non_cross`) defined in Q2 for this part, using `torch.tensor` instead of numpy arrays. The result is normalised as stated above. Also, the gold scores are normalised to make their values in the range of `[-1, 1]`. The function used for calculating tanh calibrated cosine similarity is also being made compatible with pytorch tensors and is named `tensor_cosine` in the notebook.

Data Preprocessing: The dataset is first passed to the model to get embeddings and then we use the encoded dataset for training, validation and test.

Training: This is being done by `AdamW` optimizer with two parameters `a_non_cross`, `b_non_cross` for optimization. The optimization is being done in a batched manner with a batch size of 500 over the training dataset. The training loss function is the sum of squared absolute differences of the predicted scores and given scores. For analysis during training, metrics (correlation coefficients), loss, and current value of `a_non_cross`, `b_non_cross` are printed. Also, an appropriate initialization of parameters is taken for training, after training for multiple random initializations.

Analysis



```

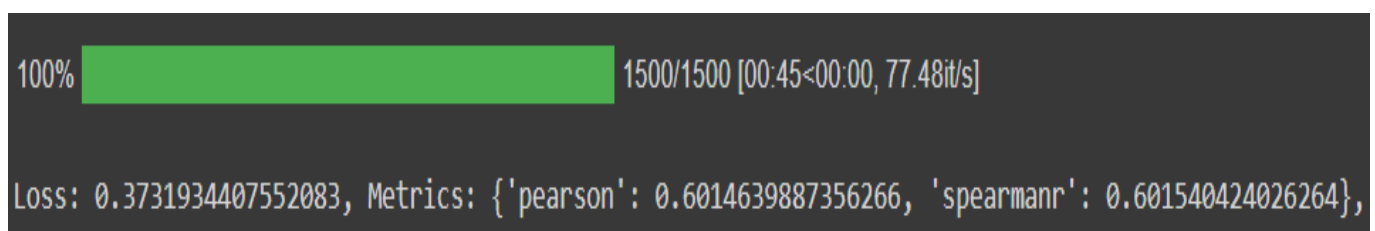
a_non_cross, b_non_cross

(Parameter containing:
  tensor(0.9762, requires_grad=True),
Parameter containing:
  tensor(-0.5234, requires_grad=True))

```

Figure 3: Trained Parameters for Non Crossing

The following is the loss, pearson coefficient and spearmanr coefficient for validation set for the above parameters:



```

100% 1500/1500 [00:45<00:00, 77.48it/s]

Loss: 0.3731934407552083, Metrics: {'pearson': 0.6014639887356266, 'spearmanr': 0.601540424026264},

```

Figure 4: Validation Set results for Non Crossing

Q3.2: Crossing

We used a new function (name: `dtw_cross`), using `torch.tensor` instead of numpy arrays. In this function, we are calculating the positive similarities components of words and adding them, to create similarity score. The result is again normalized as stated above, i.e, by dividing the score with the minimum length of two sentences. Also, the gold scores are normalised to make their values in the range of $[-1, 1]$. The function used for calculating tanh calibrated cosine similarity is also being made compatible with pytorch tensors and is named `tensor_cosine` in the notebook.

Training: Again, This is being done by **AdamW** optimizer with two parameters `a_cross`, `b_cross` for optimization. The optimization is being done in a batched manner with a batch size of 500 over the training dataset. The training loss function is the sum of squared absolute differences of the predicted scores and given scores. For analysis during training, metrics (correlation coefficients), loss, and current value of `a_cross`, `b_cross` are printed. Also, an appropriate initialization of parameters is taken for training, after training for multiple random initializations.

Analysis

```
a_cross, b_cross  
  
(Parameter containing:  
 tensor(0.9778, requires_grad=True),  
 Parameter containing:  
 tensor(-0.5218, requires_grad=True))
```

Figure 5: Trained Parameters for Crossing

The following is the loss, pearson coefficient and spearmanr coefficient for validation set for the above parameters:

```
100% ██████████ 1500/1500 [00:30<00:00, 193.84it/s]  
  
Loss: 2.137397786458333, Metrics: {'pearson': 0.537575188542166, 'spearmanr': 0.5932069535000509},
```

Figure 6: Validation Set results for Crossing

Q4: Analysis

Most of the overall analysis on validation and test datasets, are mentioned in the earlier sections and also the correlation coefficients are given. Some of the validation sets examples for all three approaches are as follows:

- Q1 - BERT Fine tuned:

```
The cougar is chasing the bear., A cougar is chasing a bear., 3.2389591641038336
The man cut down a tree with an axe., A man chops down a tree with an axe., 3.9266876971518965
The man is playing the guitar., A man is playing a guitar., 2.9219257212105036
A man is finding something., A woman is slicing something., 3.3233838600023624
The girl sang into a microphone., The lady sang into the microphone., 3.719275488561257
```

Figure 7: Only Fine Tuned BERT Tiny Model Outputs

- Q3.1 - Non Crossing:

```
The cougar is chasing the bear., A cougar is chasing a bear., 3.9801136181535037
The man cut down a tree with an axe., A man chops down a tree with an axe., 3.6195377302959004
The man is playing the guitar., A man is playing a guitar., 3.9819857398100287
A man is finding something., A woman is slicing something., 3.79858193792734
The girl sang into a microphone., The lady sang into the microphone., 3.8736319379101056
```

Figure 8: Non Crossing Approach

- Q3.2 - Crossing:

```
The cougar is chasing the bear., A cougar is chasing a bear., 3.344614952802658
The man cut down a tree with an axe., A man chops down a tree with an axe., 3.3707044273614883
The man is playing the guitar., A man is playing a guitar., 3.287190720438957
A man is finding something., A woman is slicing something., 3.229583203792572
The girl sang into a microphone., The lady sang into the microphone., 3.238288536667824
```

Figure 9: Crossing Approach

Note: We are getting good similarity scores, close to 4, in non crossing method, for similar sentences, whereas the performances of other two approaches are slightly less than the non crossing approach. We can also find an alignment using Q2, which will result in a pretty good match. Also, implementing DP approach improved the performance over fine tuned BERT.

Conclusion

DTW With Non Crossing significantly outperforms the baseline. Although, Tanh Calibration in DTW results in a lower spearmanr and pearson score over the baseline (fine-tuned BERT-tiny). But DTW with Non crossing gives more relevant score than other approaches.

Whereas, crossing approach takes more similarity for a pair of sentences than it is already there, hence for some pairs we get more than required similarity score, which inturn increases loss.