RichRep

Does an additional contrastive loss
help supervised training on text?

Group Members: Yu Xin, Rish, Marcus, Thomas, Vignesh, Rahul

Table of contents

Introduction

This project aims to clarify whether adding a contrastive learning signal offers any inherent benefit over the vanilla supervised learning setup. When given a dataset with features and labels, we recommend using an extra contrastive learning loss signal that allows the model to learn a richer, more encapsulating representation compared to the same model trained simply on the supervised signal. We hypothesise doing the former leads to highly performant models.

The dataset we are using is Quora Question Pair from GLUE Benchmark. We are using the dataset from GLUE Benchmark which gives us the train-test-val splits in .tsv format, which makes it super easy to open via pandas.

Exploratory Data Analysis (EDA)

View more of our analysis in this notebook:
https://colab.research.google.com/drive/1XkItLhHa7F2hl-YXtrgHHQlcgS4Xf16w?usp=sharing

Data Split

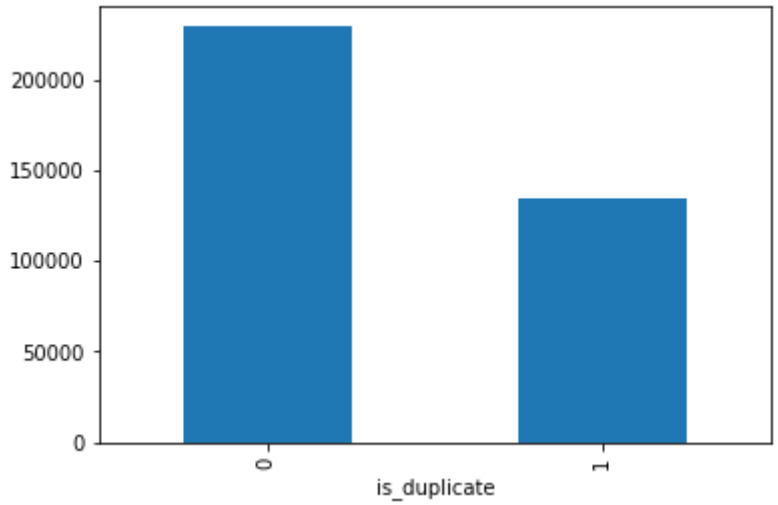


Fig 1: Dataset split

Based on our analysis in fig 1, the split between duplicate and non-duplicate data is fairly evenly distributed at 63% non-duplicate and 37% duplicate data.

Accuracy of dataset labels

Given that the Quora dataset is manually labelled, we would like to find out the accuracy of the labelling. We took a sample of 20 data points and manually compared the 2 sentences. Using manual comparison, we gauged the accuracy of the labels. We conducted this multiple times and achieved an accuracy of around 80%. However, we would like to state that this is not a very scientific way of figuring out the accuracy of the labels but it is sufficient as a gauge of the label reliability of the dataset.

Cosine Similarity of Stemmed and Unstemmed Sentences

One hypothesis tested for our EDA is the suitability of sentence embeddings as a relevant feature for classification. Sentences are processed with SentenceBERT to form vector representation. Cosine similarity between duplicate and non-duplicate sentences was measured.

It is believed that different stemming procedures can have an effect on the cosine similarity of vector representations. As a test of this hypothesis, cosine similarity of Porter-stemmed sentences and Lancaster-stemmed sentences was compared with the cosine similarity of unstemmed sentences.

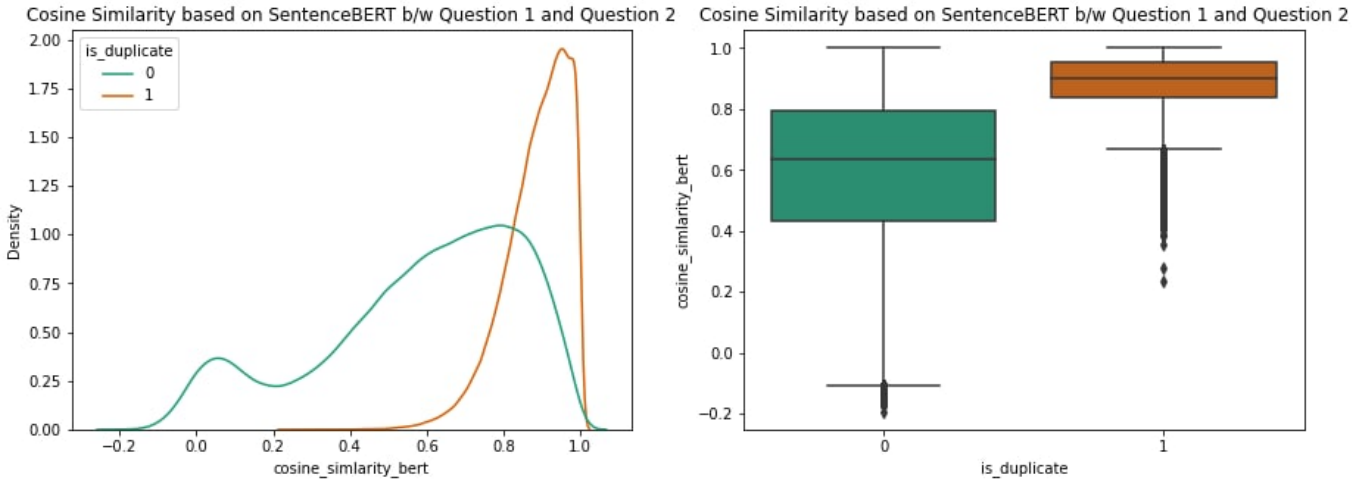


Fig 2: Cosine Similarity on Raw Original Sentence Embeddings

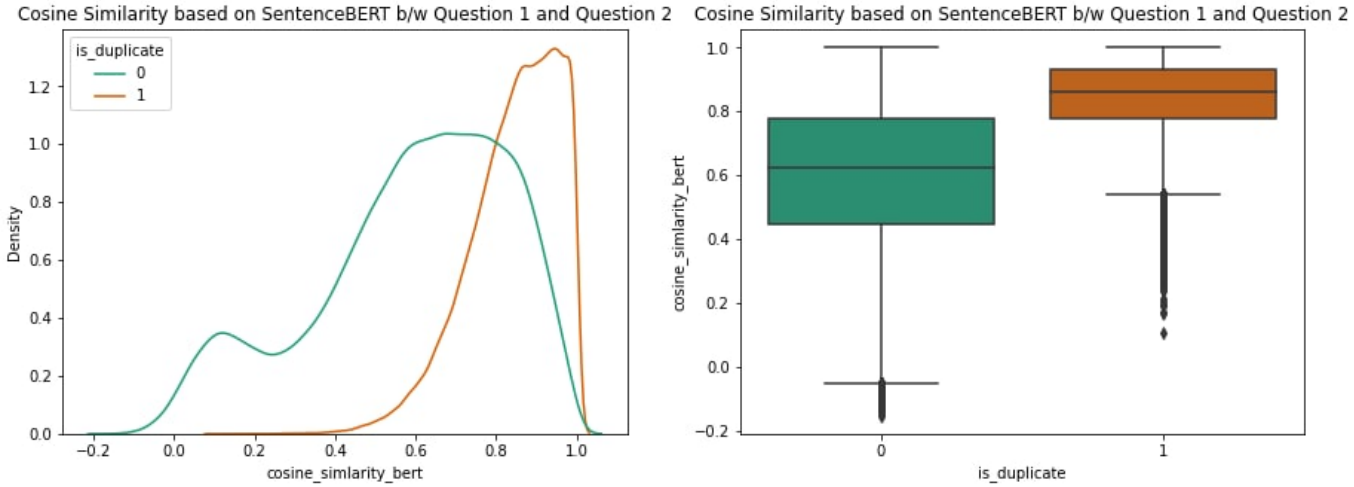


Fig 3: Cosine Similarity on Porter-Stemmed Sentence Embeddings

From Fig 3, the difference in cosine similarity is not as significant as in raw sentence embeddings in fig 2. However, cosine similarity is still a useful feature to determine whether a pair is a duplicate or not. We can see that there would be significantly different values in cosine similarity of duplicate and non-duplicate pairs.

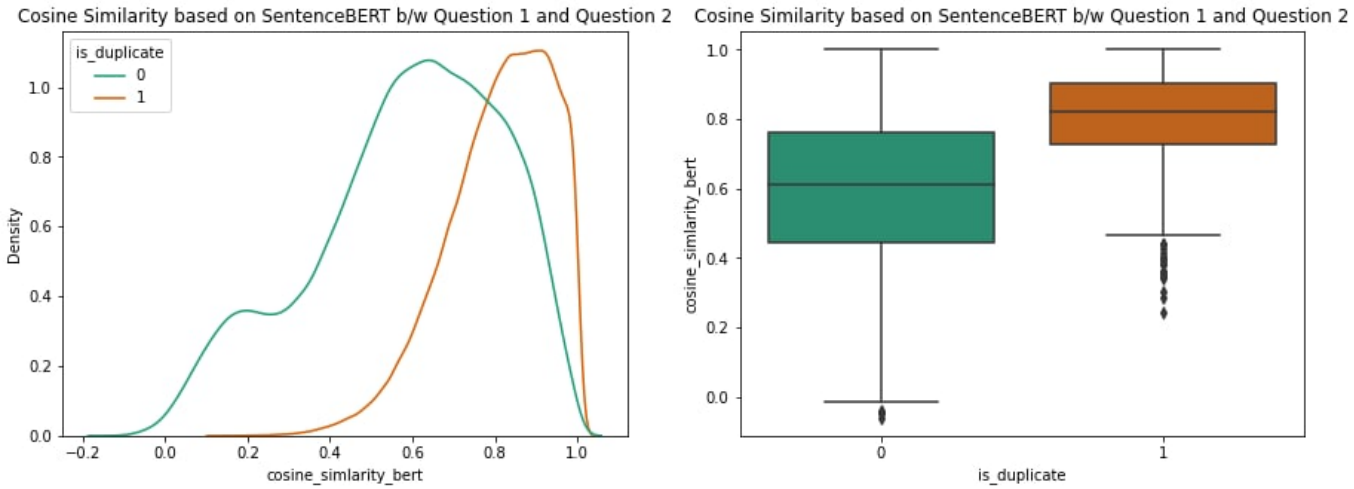


Fig 4: Cosine Similarity on Lancaster-Stemmed Sentence Embeddings

We conducted the same experiment using Lancaster-Stemmed Sentence Embedding and got a worse result than Porter-Stemmed Sentence Embeddings.

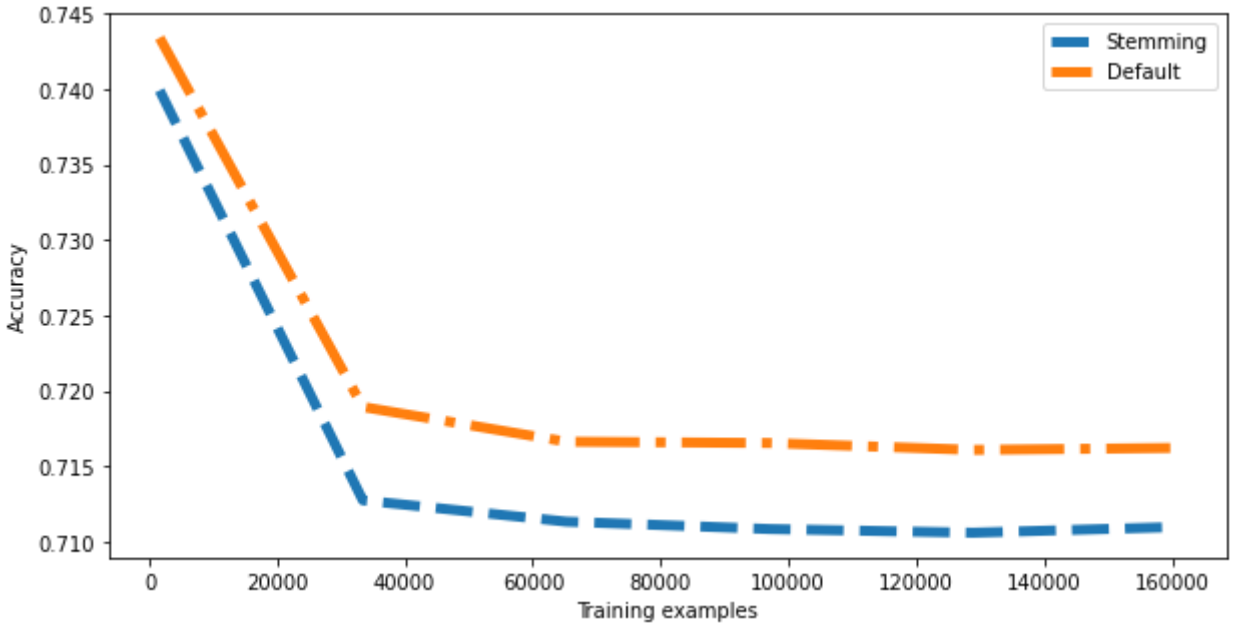
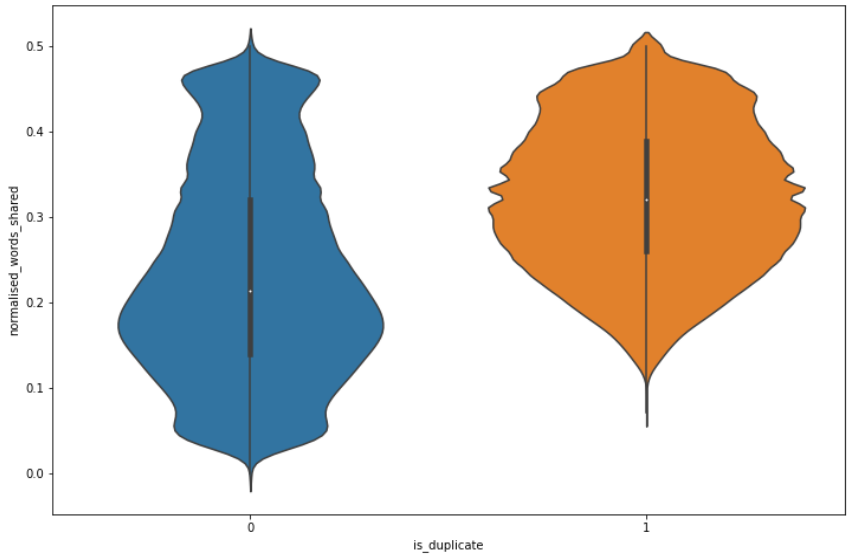


Fig 5: Result from stemming sentence

Fig 5 shows the training result of using Lancaster Stemmer. It is evident that the stemming procedure showed worse results than unstemmed sentences. This is because stemming reduces the amount of information, making one stem for the different word forms. Lancaster Stemmer is especially aggressive and can tend to overstem words. This can bring noise to the model since some stems of the different words can be ambiguous, and the model can't be able to separate "playstation" and "play".

Similarity of words between sentences

Another hypothesis tested during EDA is that Bag-of-Words (BOW) representation of sentences can provide meaningful indication of similarity of sentences through the number of shared words, measured by set similarity. Set similarity was calculated with the size of intersection / (size of first set + size of the second set). Higher set similarity indicates that the sentences share more words, and are more similar.



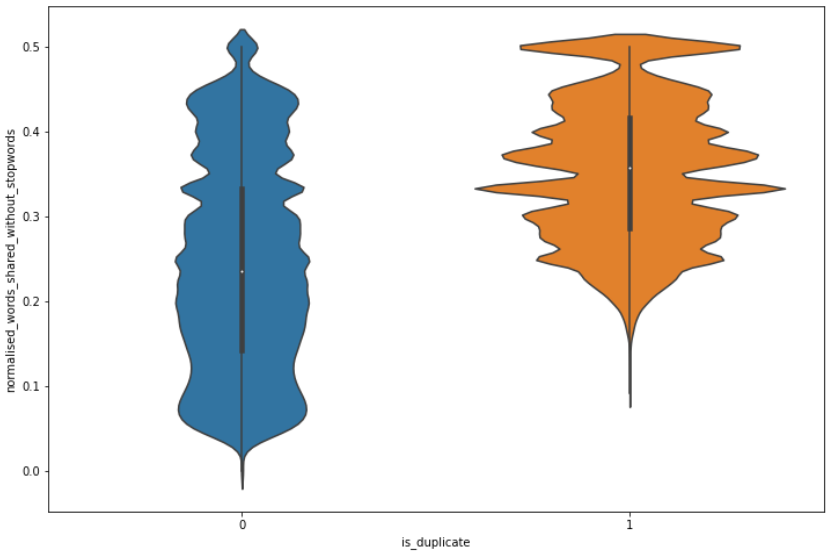
	Not duplicate	Duplicate
Mean	0.232835	0.323786
25% percentile	0.139535	0.260870
75% Percentile	0.320000	0.388889

Fig 6: Percentage of similar words within sentence

Based on the result in Fig 6, we note that:

- Mean of duplicate and non-duplicate is 0.090951 away from each other
- Mean of duplicate lies 0.003786 above the 75th percentile of non-duplicate
- Mean of non-duplicate lies 0.028035 below from the 25th percentile of duplicate

As an extension of our hypothesis, we wondered if removing stop words will decrease the number of noninformative words, contributing to a more meaningful BOW representation containing only semantically relevant words. This is hypothesized to decrease the number of shared words between non-duplicate sentences, resulting in a greater difference in set similarity metric between duplicate and non-duplicate sentences.



	Not duplicate	Duplicate
Mean	0.245569	0.353381
25% percentile	0.142857	0.285714
75% Percentile	0.333333	0.416667

Fig 7: Percentage of similar words in sentences without stopwords

From the result in Fig 7, we note that:

- Mean of duplicate and non-duplicate is 0.107812 apart
- The mean of duplicate lies 0.020048 lies 75% above non-duplicate
- The mean of non-duplicate lies 0.040145 below the 25%

By comparing the results before and after removing of stopwords, we can see that removing of stopwords will give an increased difference in mean of the set similarity metric between duplicate and non-duplicate sentences. This indicates that the removal of stop words is a good preprocessing step to increase the usefulness of the set similarity metric. However, we decided not to use this as a preprocessing step as it might remove information from the sentence that might be crucial for our model. Words such as not, no, never and unless negates the semantic meaning intended but might be removed by classic stop words removal lexicon. We decided to keep these stop words to preserve contextual information present within the sentences which might be extractable using more powerful preprocessing models (Eg. BERT).

Model Design

Our guideline for the project is “supervised learning + contrastive learning must be used in conjunction to reap the benefits in both small and large dataset regimes”. As such, our experiments are crafted based on the guidelines

Architecture Diagram

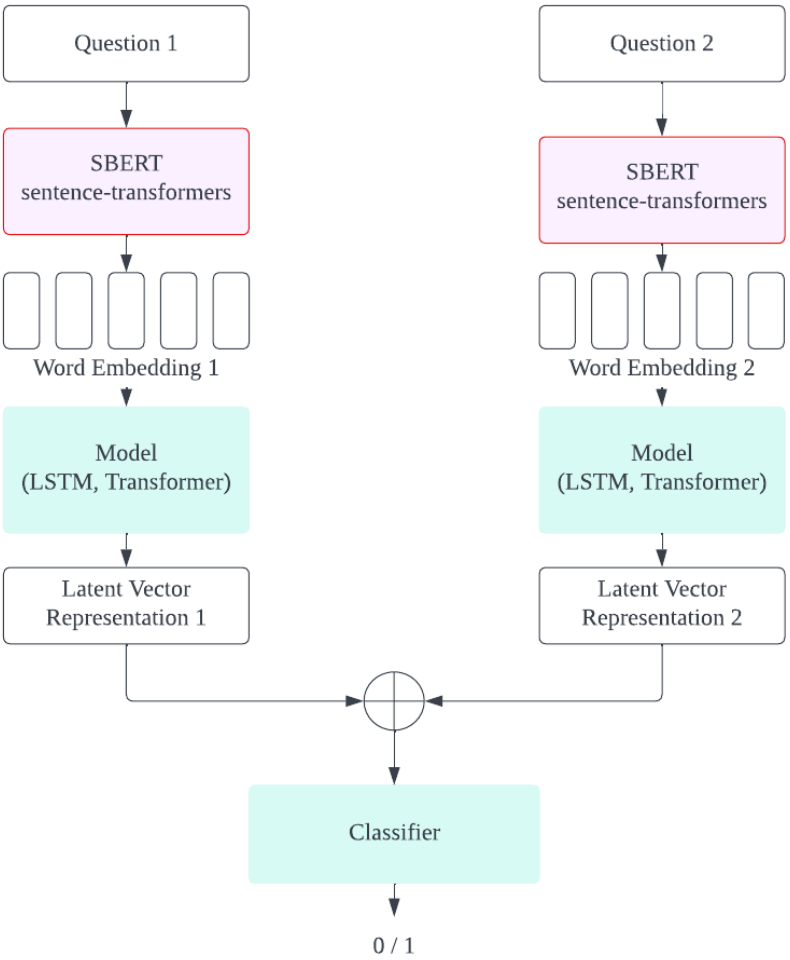


Fig 8: Architecture diagram of our model

Firstly, we made use of SBERT sentence-transformer to transform each question into a sentence embedding. The sentence embedding is then passed through our model (Transformer/LSTM) to get a latent vector representation for each model. These latent vector representations are concatenated and finally passed into our classifier where we will receive an output of either 0 or 1, representing non-duplicate and duplicate classification respectively.

Transformer Model

Apart from the standard LSTM model, we are also testing our hypothesis using Transformer. The transformer relies solely on self-attention mechanism, popularised by (Vaswani et al., 2017). The mechanism is a pairwise scoring function that considers relative importance of N tokens to the context of the sequence.

The self-attention mechanism is formulated as follow, given a sequence of embedding $X \in \mathbb{R}^{T \times d}$,

$$\mathrm{A} = \mathrm{softmax}\left(\frac{XW_Q(XW_K)^T}{\sqrt{d}}\right)$$

$$\mathrm{head}_i = \mathrm{Attention}(XW_Q, XW_K, XW_V) = A(XW_V)$$

$$\mathrm{MHSA}(X) = \mathrm{concat}(\mathrm{head}_1, \mathrm{head}_2, \cdots, \mathrm{head}_H)W_O$$

where W_Q, W_E, $W_v \in \mathbb{R}^{d \times d}$ are trainable parameters, and d is the projection subspace dimension of the attention operation.

The elements a_{ij} of the attention matrix A are pairwise scores on how important token i perceives j to be to the context of the sequence. This self-attention operation can abstracted into a "head", and when some H number of heads perform the operation in parallel for the same input sequence X, with independent parameter matrices W_Q, W_K, W_v, it is called multi-head self-attention (MHSA). The outputs of the different heads h are concatenated and projected using trainable parameter matrix $W_o \in \mathbb{R}^{Hd \times d}$.

Hypothesis 1: CL loss helps to improve model performance

Hypothesis: An additional CL loss helps improve the performance of models.
To verify: SL+CL model outperforms SL model on QQP dataset.

LSTM result

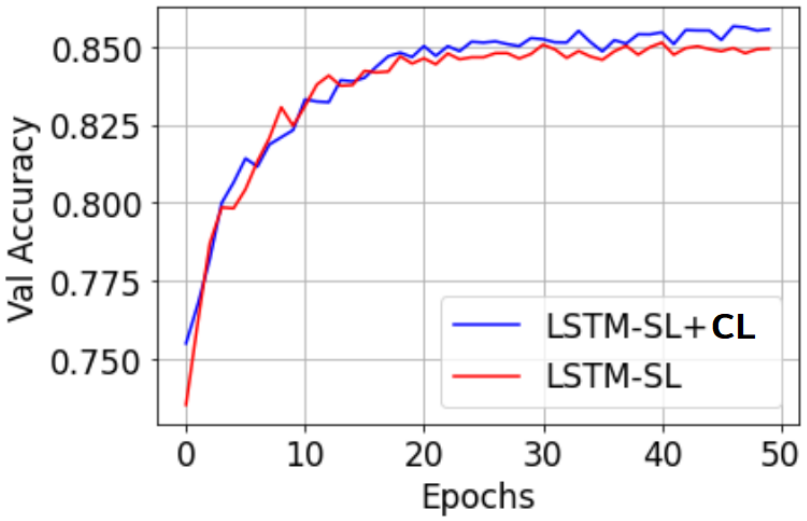


Fig 9: Val accuracy-epoch curve for LSTM SL and SL+ CL

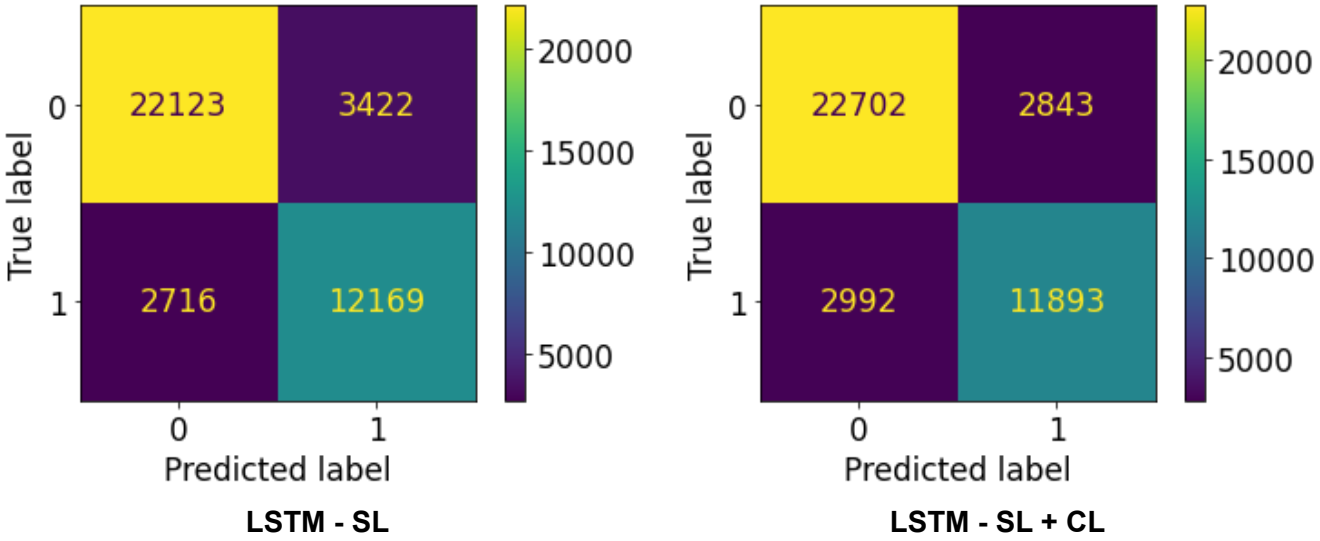


Fig 10: Confusion Matrix for LSTM Model

As observed from Fig 8, the model trained with LSTM-SL LSS loss consistently shows higher validation accuracy than the model trained with LSTM-SL loss after 20 epochs. At the end of the training, confusion matrix for the model trained with LSTM-SL loss and LSTM-SL+CL loss is shown in Fig 9.
The model trained with LSTM-SL loss has a precision of 0.866, recall of 0.891, and accuracy of 0.848.
The model trained with LSTM-SL+CL loss has a precision of 0.889, recall of 0.884, and accuracy of 0.856.

Transformer result

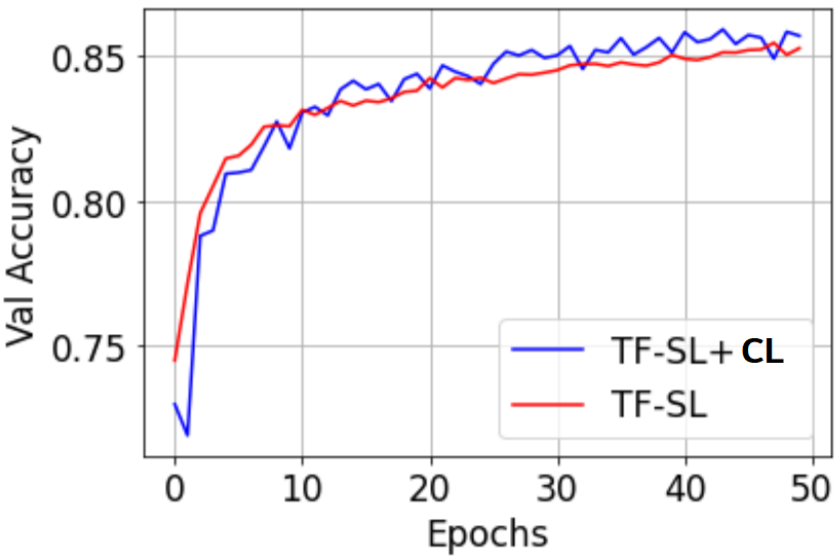


Fig 11: Val accuracy-epoch curve for Transformer SL and SL+CL

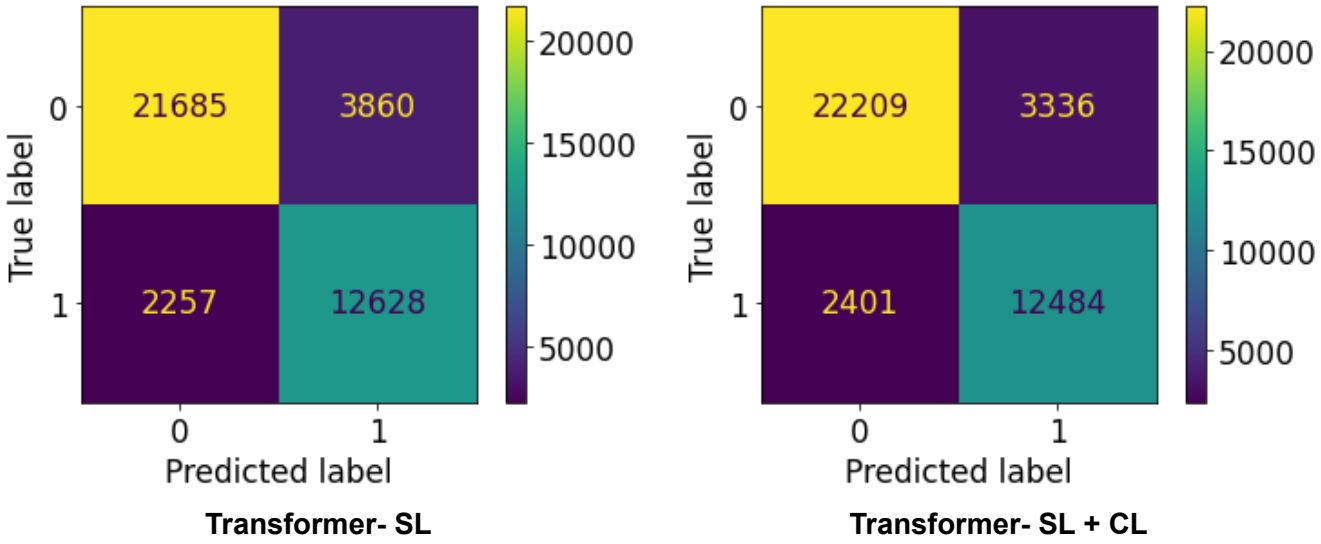


Fig 12: Confusion Matrix for Transformer Model

The benefits of using CL loss are similarly observed with transformers. As shown in Fig 10, the model trained using CL resulted in higher validation accuracy. The confusion matrix for models trained with Transformer-SL and Transformer-SL+CL is shown in Fig 11. The model trained with Transformer-SL loss has a precision of 0.849, recall of 0.906, and accuracy of 0.849. The model trained with LSTM-SL+CL loss has a precision of 0.870, recall of 0.902, and accuracy of 0.858.

Experiments with scale

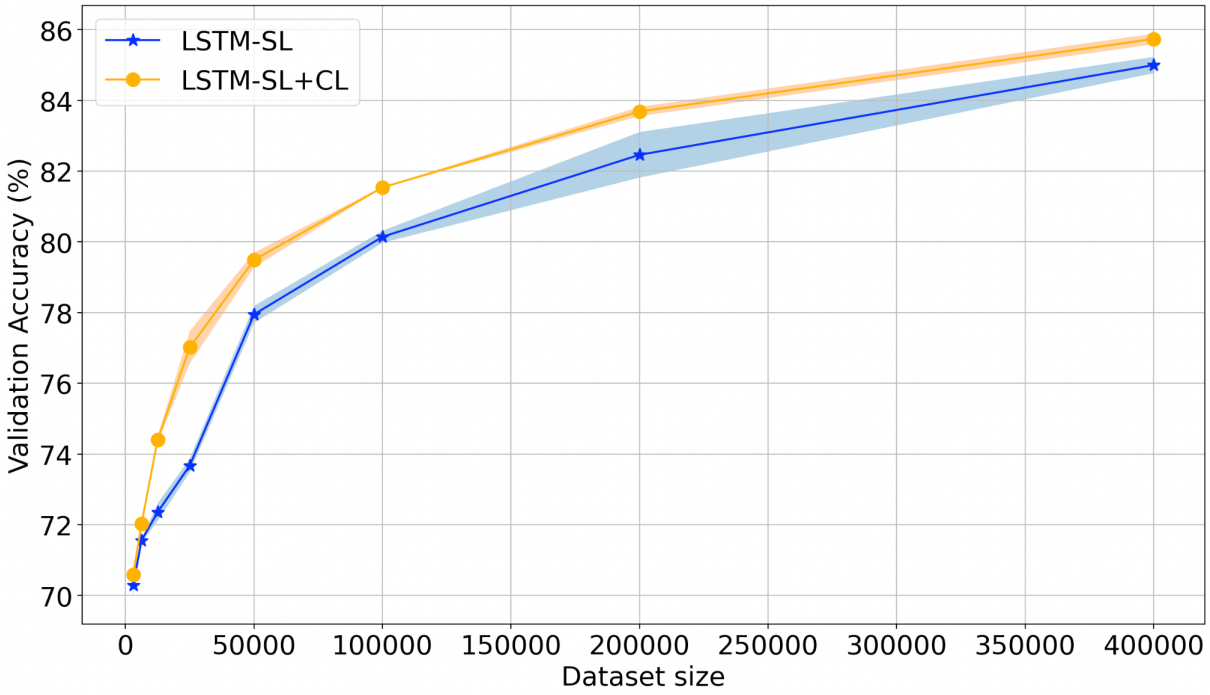


Fig 13: Performance of LSTM model over different training set sizes

To find out how the benefits of using CL loss vary with the size of the training set, models are trained with an exponentially increasing amount of data. Fig 12 shows that models trained with LSTM-SL+CL loss dominate LSTM-SL loss for all sizes of the training set used.

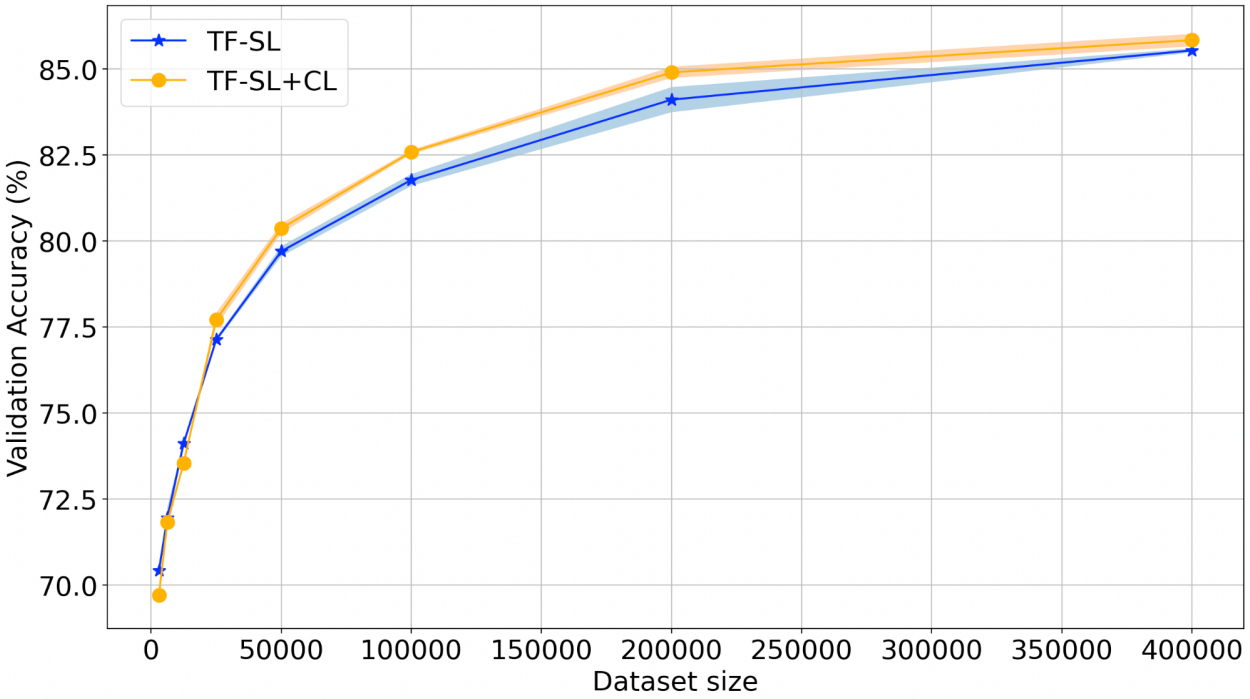


Fig 14: Performance of Transformer model over different training set sizes

Based on the result of Fig 13, we are also able to obtain a similar trend for the transformer.

Hypothesis 2: SL+CL model learns some extra “tacit” knowledge during training, which correlates with better model performance.

For this hypothesis, we choose our simplest but performant model, the LSTM, for further ablation on performance. Ultimately, we hope to make some educated guesses on WHY the CL-augmented model does better.

This section is covered more in depth in the video and only a proportion of the results is shown in the appendix.

Centralised Kernel Alignment

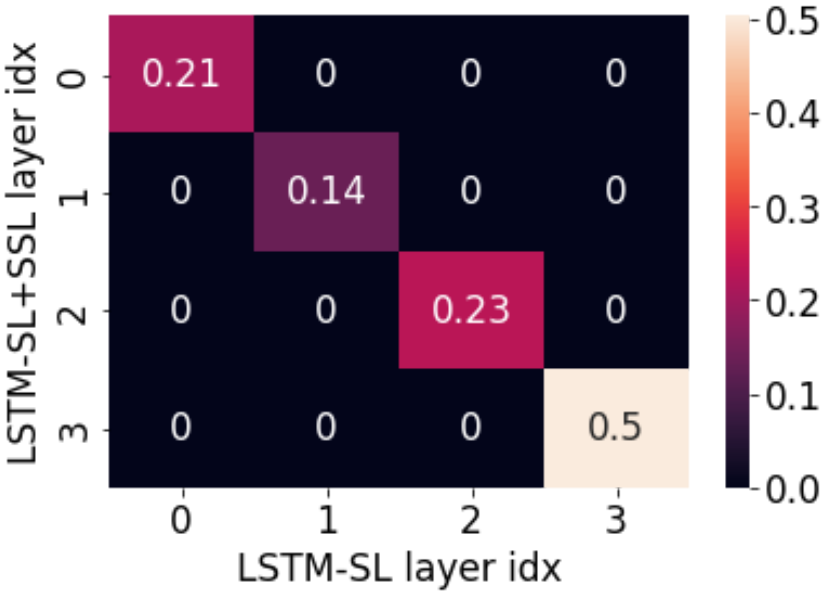


Fig 15: CKA result

Centralised Kernel Alignment is used to measure the global similarity of representations of two models. A score between 0 to 1 will be generated. Scores near 0 will mean models learned different things from data whereas scores near 1 mean models learned similar things from data.

From the result we obtained in fig 15, shallow layers from both models are more dissimilar while deeper models are more similar. Hence, we identify that SL and SL+CL models learn very differently on the same dataset

Hypothesis 3: The CL loss augments the training process for models but also needs the SL loss to be there

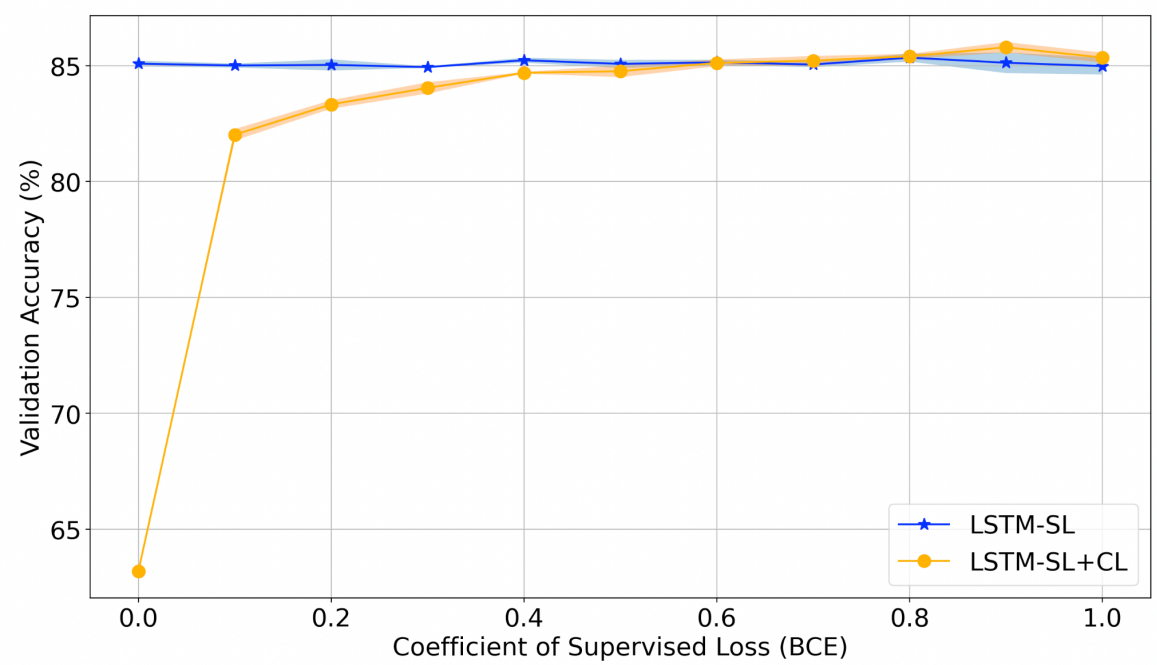


Fig 16: Validation - SL loss coefficient graph for LSTM

By varying the percentage of SL loss components during training, we are able to determine the importance of SL loss for training. Below 0.7 SL loss coefficient, LSTM-SL+CL mode validation accuracy consistently underperformed LSTM-SL model validation accuracy. The performance of the LSTM-SL+CL model steadily declined with decreasing SL loss component, with a sharp dropoff at 0 SL loss coefficient. As such, we can observe that SL+SLL must be used in conjunction to reap the benefits of CL loss.

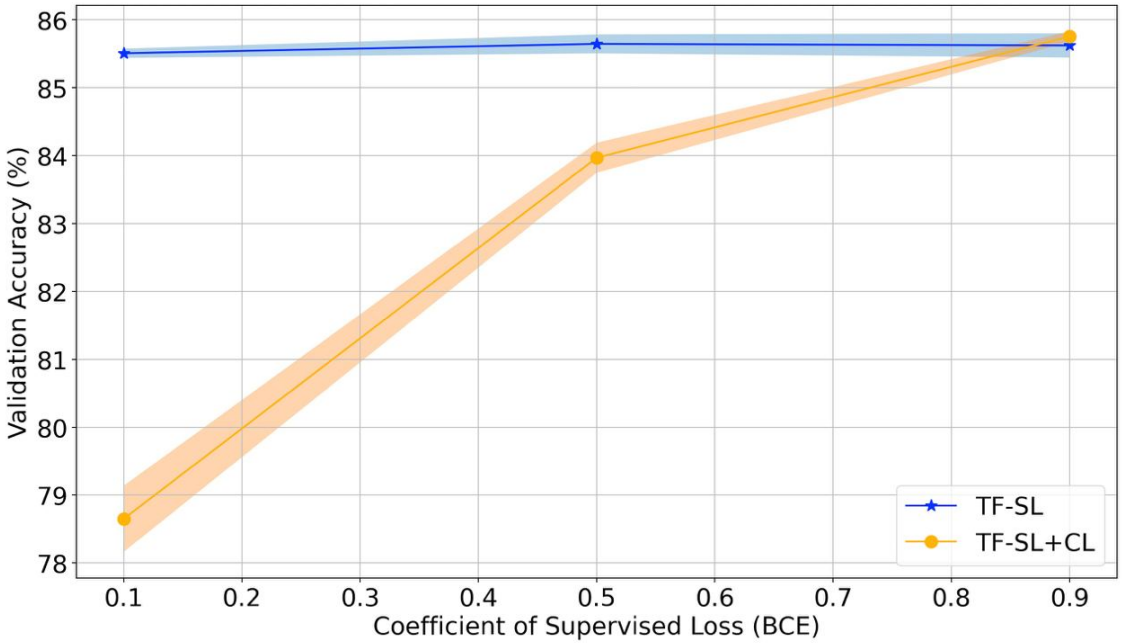


Fig 17: Validation - SL loss coefficient graph for Transformer

From the transformer graph above, we are able to see a similar trend where the transformer with SL+CL outperformed SL when there is a coefficient of 0.9.