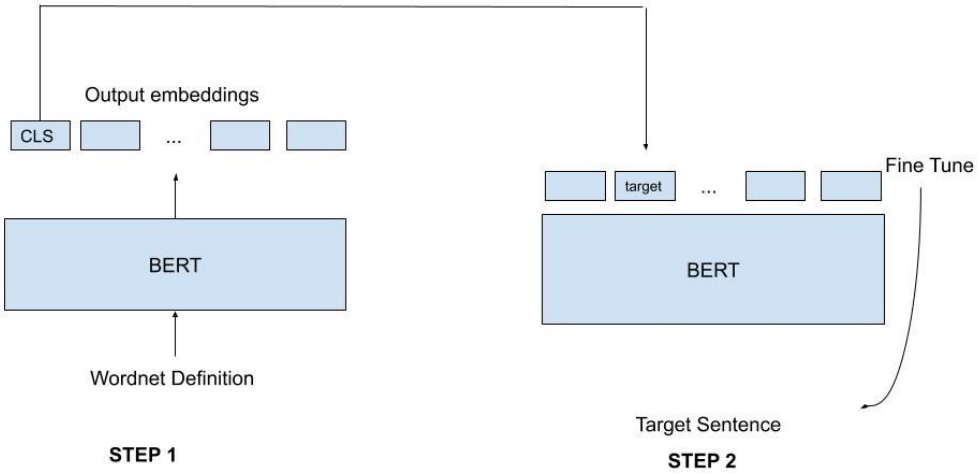
Approaches

We take two main approaches.

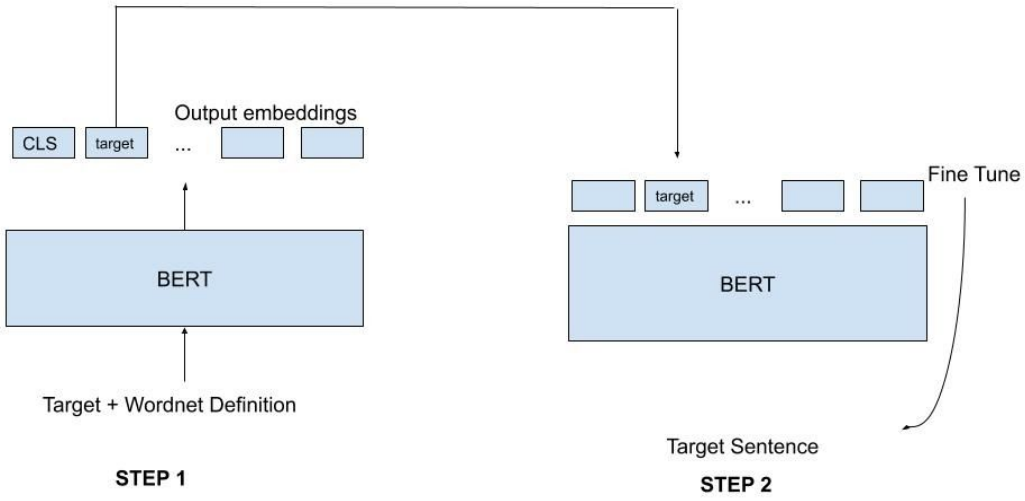
- We fine tune the target word in each training sentence based on the CLS token of the sense definition
- We append the target word to the beginning of the sense definition. We then use the embedding of the target word inside the definition to fine tune the target word in each sentence of the training data.

Both approaches are shown in the diagrams below:

Fine Tune based on CLS of Wordnet Definition



Fine Tune based on target token (attached to Wordnet Definition)



After fine tuning the BERT model, we obtain all instances of the words: *right, back, left, serve, open* from Semcor. We then do a train/test split and build a NN classifier for each word. This is identical to the model architecture from a few weeks ago.

Results

Neural Network Accuracy on Test Set

Model Type	*Serve*	*Back*	*Left*	*Right*	*Open*
BERT-Base (original)	0.69	0.81	0.66	0.91	0.75
BERT-Base (Fine-Tuned on CLS) batch_size=8, epoch=1	0.35	0.40	0.46	0.32	0.26
BERT-Base (Fine-Tuned on CLS) batch_size=16, epoch=4	0.33	0.53	0.49	0.56	0.37
BERT-Base (Fine-Tuned on target word concatenation) batch_size=16, epoch=4	0.35	0.35	0.32	0.24	0.26

Analysis

Raising the number of cycles (epochs) and batch size seems to increase sense information in the fine-tuned embeddings.

The CLS token also proves to be a better representation of a target word's sense than the second method (appending word to definition). Overall, accuracy is lower for every word than using BERT-Base original embeddings.

Implementation

We imported the BERT model into Tensorflow using Keras. We removed the NSP and Masked Language Modeling layers and created a custom Lambda layer. This Lambda layer dynamically picks the target word to fine-tune based on the position inside the sentence where it occurs.

This means that a custom loss function does not have to be written. We use 'mean squared error' to calculate loss. We set the imported BERT model to accept an input of max length 200 tokens. We use the BERT-Base model in both fine-tuning and testing. Only the last embedding layer is used (as opposed to last 4 concatenated or second to last) in testing. However, fine-tuning is conducted on all of the BERT layers.

Next Steps

1) Attempt combination of last/second-to-last/concatenate-last-four layers when extracting BERT embeddings. Last layer is typically not the most ideal.
2) Fine-tune only the last few layers, but freeze the lower layers of BERT so they remain unchanged.

3) Use BERT-Large to fine-tune and generate embeddings instead of BERT-small.

Notes

- BERT-Large is the model we used in June for this experiment. It obtained better results than using BERT-Base. Here, we switched to the BERT-Base version just for the sake of memory efficiency. However, BERT-Large should output higher accuracy