ENTERPRET ASSIGNMENT

Deepam Minda

This is the report containing the summary of the experiments I performed.

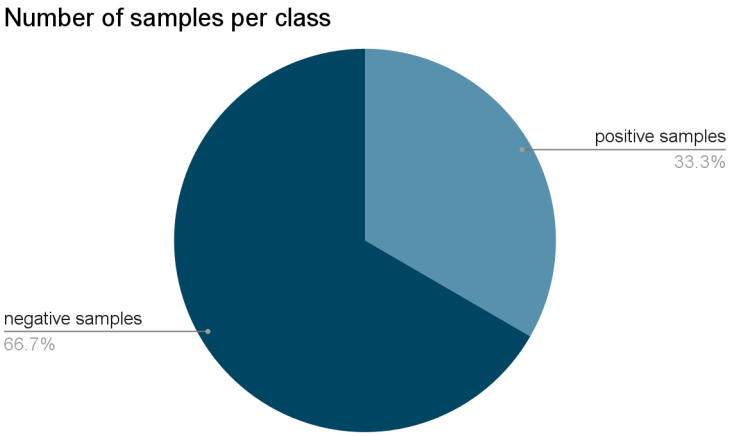
DATA INSIGHTS:

A. Distribution of labels

The training data consists of only positive samples, 2061 in total. The test data has 9000 samples,

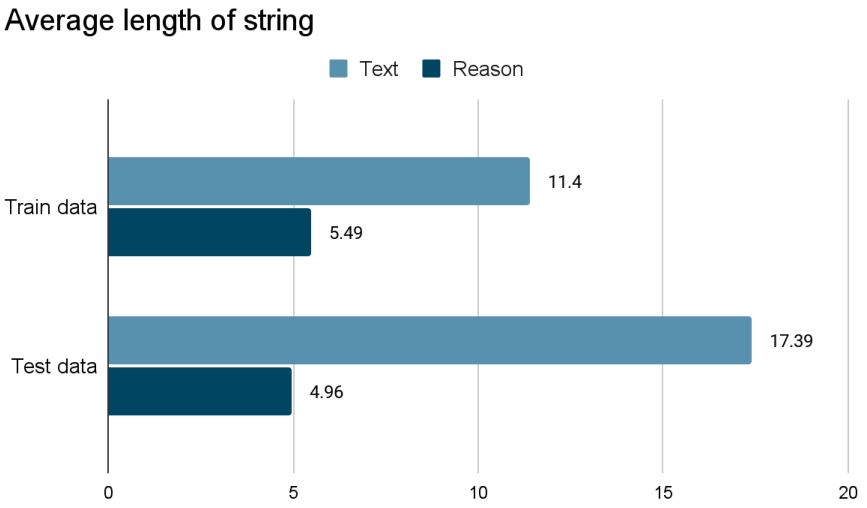
- 3001 are positive
- 5999 are negative samples.

Clearly, there is a class imbalance in our datasets. More of this is addressed in the training section.



B. Average number of words in text and reason

Next, we have a look at the lengths of texts/comments in both our features, ie **text** and **reason** for training and test data.



I also note that the test data seems to have longer sentences compared to the training data.

TRAINING APPROACH:

A. Training Approaches:

The task we have at hand is “Given a text and a reason, predict if the text satisfies the reason.” This can either be treated as one of 2 downstream tasks, Sentence Similarity or Natural Language Inference (entailment, neutral, contradiction). For SOTA performance, we have open-source pre-trained transformers available for both of these tasks, which are transformers finetuned on large relevant data. But for my assignment, I will go ahead and try to finetune a BERT with our own training data and evaluate it on the test set..

One significant difference between our test and training data is that we have only positive samples in the training data, while the test data has both positive(1) and negative(0) samples. We can either train a Bi-encoder or a Cross-encoder using any base transformer. Both can take a pair of sentences and return us a label (1/0).[see appendix for more information].

Both encoders have their advantages and disadvantages, and for deployment in production, we would try all possible avenues to optimize factors such as latency and throughput. But given our short timeframe, I have trained a bi-encoder model.

Hyperparameters:

A batch size of 32 was used for training. 10 percent of the samples were used for warmup along with a linear scheduler. Training was done for 5 epochs, with the AdamW optimizer and the best checkpoint was saved for evaluation.

Most of the choices like batch size, schedular, and optimizer were arbitrary and hyperparameter tuning would be required to select the best parameters.

B. Evaluation metrics:

I have tracked several metrics across our experiments, **accuracy, recall, precision**, and **f1.** In addition, I have used Average Precision Score as my main evaluation metric, because it summarizes the PR curve and factors into account all thresholds.
For each of these metrics, higher is better. For metrics that require a threshold, we select the threshold which gives the best accuracy.

C. Ablation Study:

1. Base Models Used:

- bert-base-uncased
- bert-large-uncased

Metric - average precision score (higher is better)

Base Model	Model file size (mb)	Model size (in million parameters)	Metric before training	Metric after finetuning with MNRLoss
bert-base-uncased	420	110	0.542	0.655
bert-large-uncased	1250	340	0.472	0.647

Despite being the smaller model, bert-base-uncased gives us marginally better performance both before and after finetuning. Since smaller models are faster to train and experiment on, we'll use this for further experiments.

2. Text inputs passed:

Instead of just passing input as (text, reason), I also passed reverse pairs (reason, text). This way we can make the most of our small dataset. This improved performance on the test set when bert-base-uncased was trained with MNR loss (explained next).

	Avg Precision	accuracy	f1	recall	precision

	Score				
Single input pairs	0.642	0.740	0.554	0.485	0.6495
Both pairs	0.655	0.746	0.546	0.458	0.674

Performance marginally improves when both pairs are used.

3. Losses Used:

- **Cosine Similarity** -

For each sentence pair, we pass sentence A and sentence B through our network which yields the embeddings u und v. The similarity of these embeddings is computed using cosine similarity, and the result is compared to the gold similarity score (which in our case is only 1.0 for train set).

- **Multiple Negatives Ranking** -

This loss only expects positive pairs and for each pair *(a,b)*, it uses all others as negative samples. For each *a*, it uses all other *b* as negative samples, i.e., for each *a*, we have 1 positive example *(b)* and n-1 negative examples *b*. It then minimizes the negative log-likelihood for softmax normalized scores.

By design, MNR loss works better than Cosine Similarity. This is reflected in our experiments. When using the cos-sim score we hardly improve the performance of the base model, while using MNR loss we gain almost 9 points in accuracy and 10 points in avg precision score.

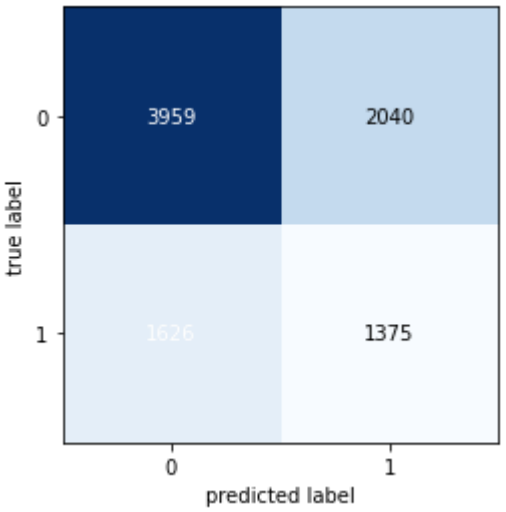
Given table compares evaluation metrics achieved after finetuning bert-base-uncased model.

Loss used	Avg Precision Score	accuracy
cos-sim	0.556	0.709
MNR	0.655	0.746

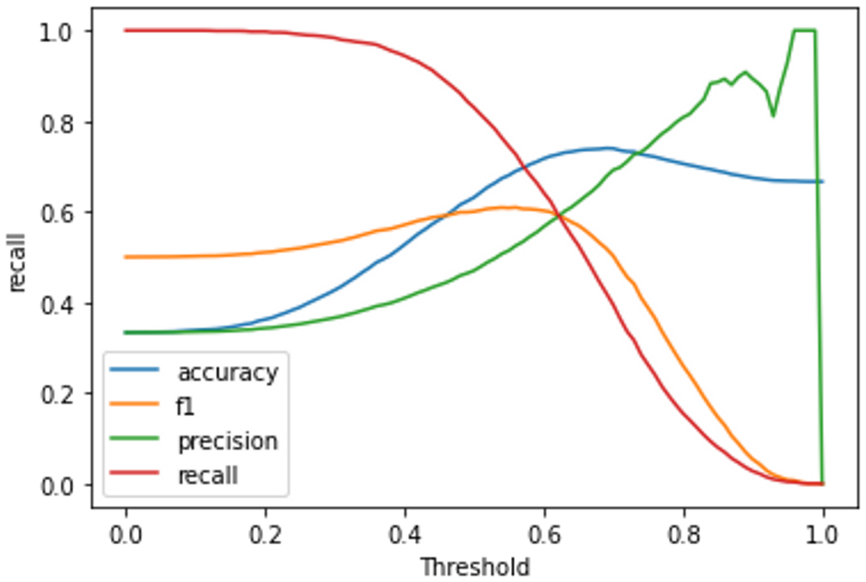
ERROR ANALYSIS:

Here is an error analysis of our best performing finetuned model,
i.e. bert-base-uncased / MNRLoss / both input pairs

- Our model achieves a **true positive rate of 45.8% and true negative rate of 65.99%.**



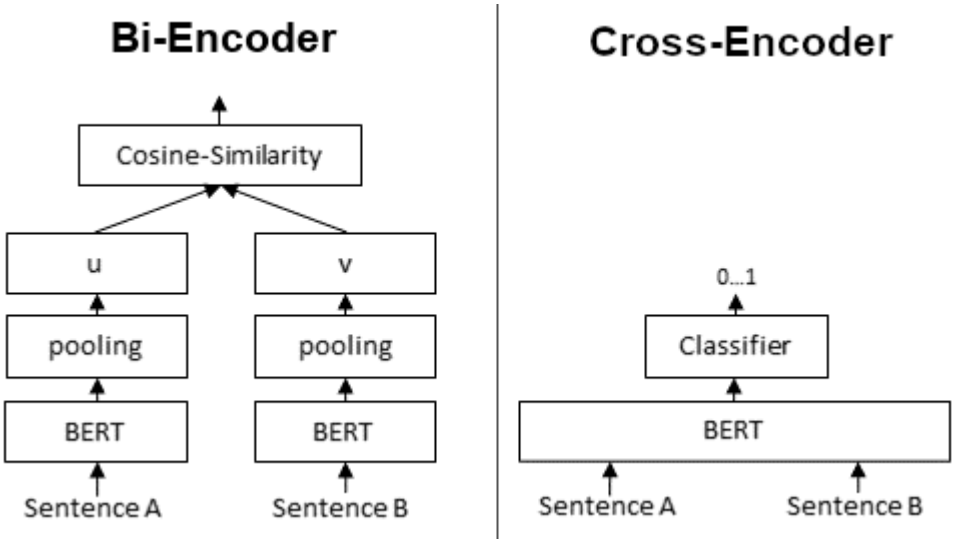
- Given below is how our model fared for different thresholds.



APPENDIX

1. Bi-encoders VS Cross-encoders

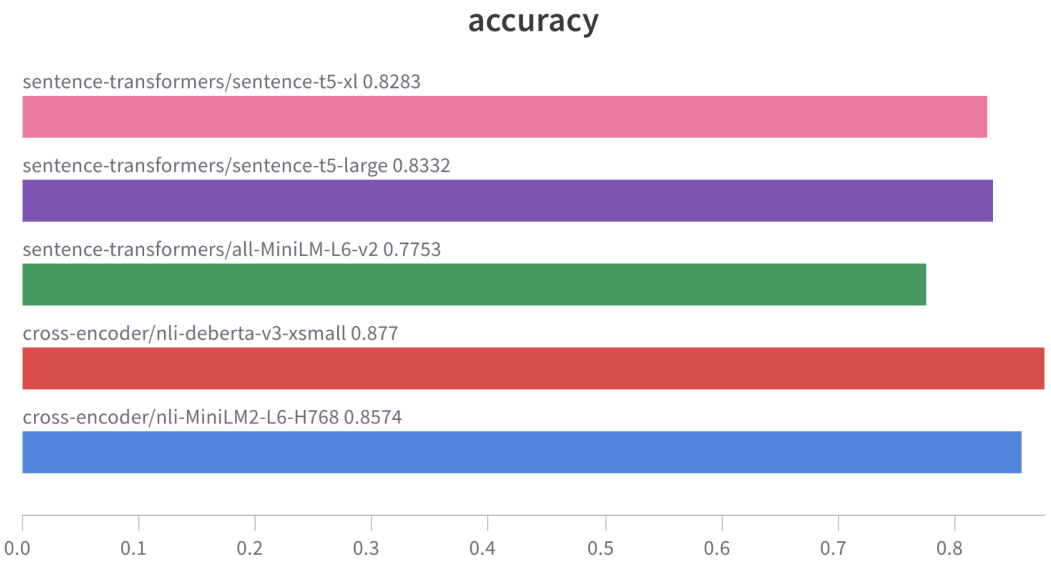
Bi-Encoders produce for a given sentence a sentence embedding. We pass to a BERT independently the sentences A and B, which result in the sentence embeddings u and v. These sentence embeddings can then be compared using cosine similarity:

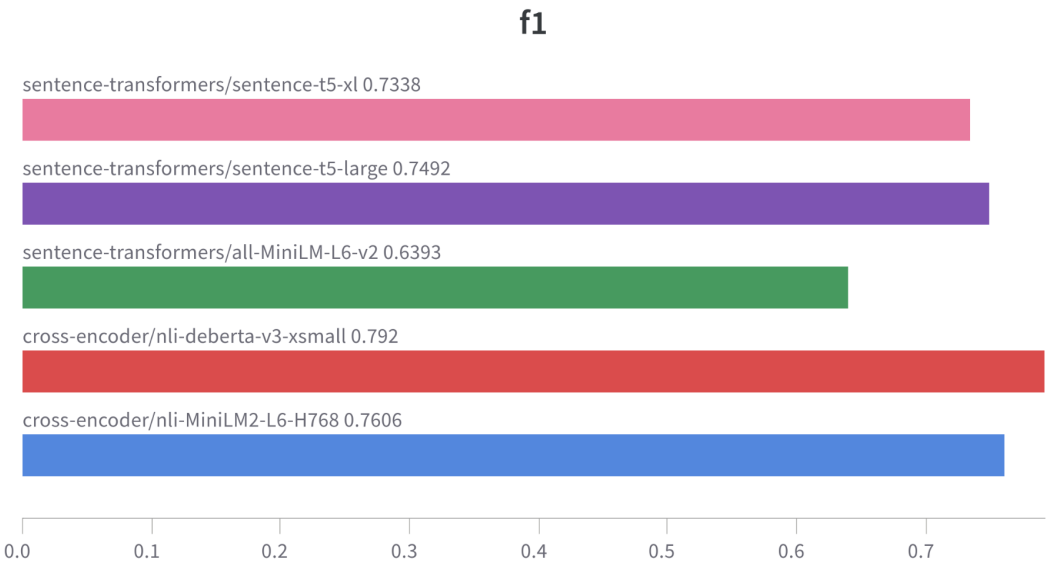


In contrast, for a Cross-Encoder, we pass both sentences simultaneously to the Transformer network. It produces an output value between 0 and 1 indicating the similarity of the input sentence pair:
A Cross-Encoder does not produce sentence embeddings. Also, we are not able to pass individual sentences to a Cross-Encoder.
Cross-Encoders achieve better performances than Bi-Encoders. However, for many applications, they are not practical as they do not produce embeddings we could e.g. index or efficiently compare using cosine similarity.

2. SOTA

Due to the size of our training data we can not achieve SOTA in NLI, sentence similarity, or any downstream task. The answer is open-source sentence transformers which are trained on huge datasets like NLI, SNLI, MultiNLI, STSbenchmark, etc.
Just for reference, this is the performance of a few SOTA models on our test dataset.





As we can see, these models give the best accuracy of **87.7%** (nli-roberta-v3), while we achieved only **74.6%** accuracy.