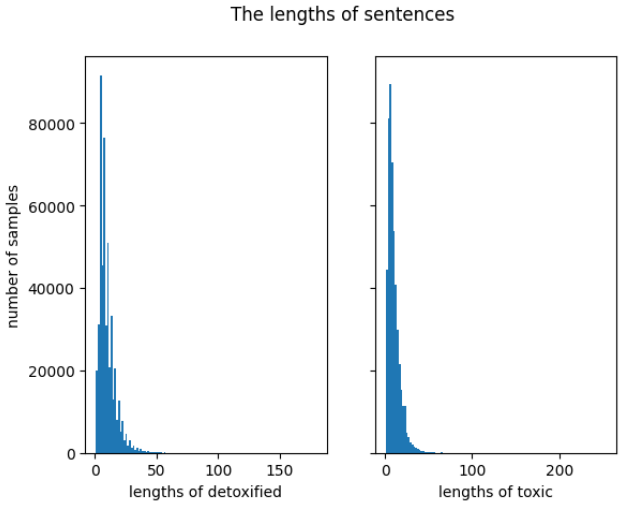
Final Solution Report

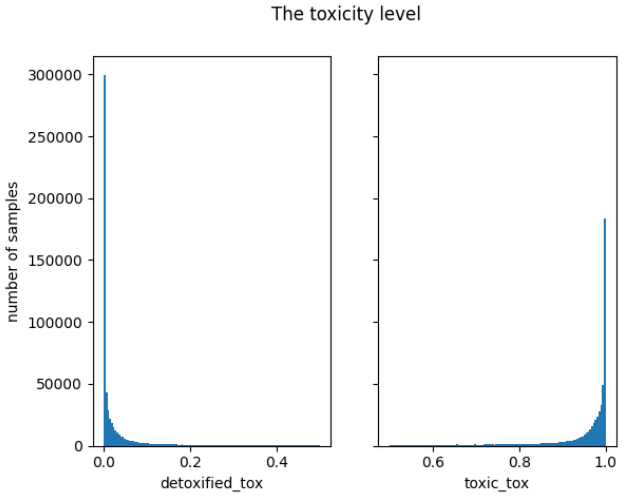
Introduction

The primary aim of the assignment was to create an algorithm, model, or a group of models that can transform text with a toxic style into text with a neutral style while retaining the same meaning. There exist several methods for addressing this challenge, including dictionary-based approaches and machine learning models. I chose to investigate different Seq2Seq models that rely on deep neural networks. In my work, I implemented and assessed three distinct models: LSTM, AutoEncoder LSTM, and Transformer. Additionally, I conducted experiments using the T5-base pre-trained language model for the detoxification task.

Data analysis



(figure 1)



(figure 2)

The dataset used for model training, called ParaNMT, includes pairs of reference and translation texts along with their toxicity levels. The analysis of this dataset reveals the following findings:

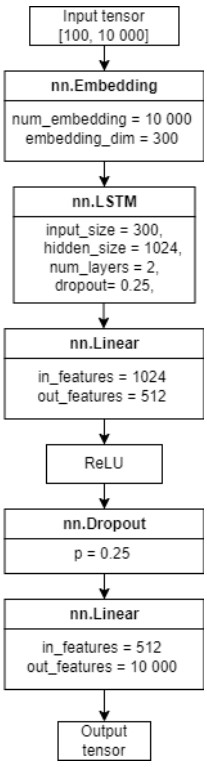
1. Some translation texts are not less toxic than the reference texts.
2. In some pairs, both the reference and translation texts can have either high or low toxicity levels.
3. The majority of texts in the dataset consist of fewer than 100 words (fig. 1).

Based on these findings, it was decided to add *toxic* and *detoxified* columns based on the toxicity level. I then plotted the distribution of toxicity levels in these two columns, and it became evident that the majority of texts fell within the range of 0.2 to 0.8 in terms of toxicity (fig. 2). Consequently, texts falling outside of this range were removed, as well as overly lengthy texts with more than 100 words.

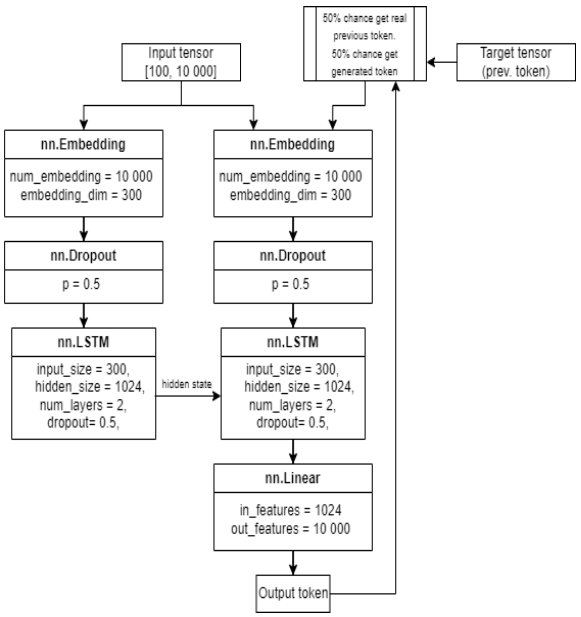
Model specification

All the models I tested were Seq2Seq models, and they varied in architecture, but they were all based on the encoder-decoder pattern (fig. 3,4). The only exception was the first model, which was a simple LSTM and did not follow this pattern (fig. 5).

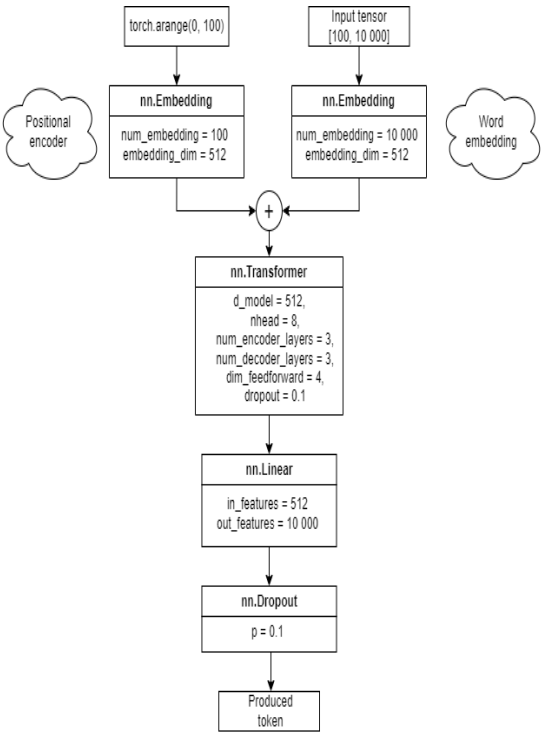
The architecture for the Transformer model was inspired by this video and this PyTorch tutorial, while the AutoEncoder LSTM architecture drew inspiration from this video.



(figure 3)



(figure 4)



(figure 5)

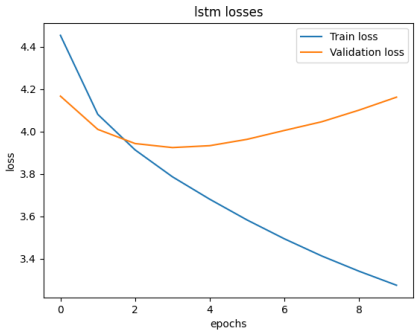
Training process

I encountered various challenges during the training process. First and foremost, the vocabulary size was excessively large, resulting in extended training durations even for simple models. Additionally, the dataset itself was quite extensive, requiring a

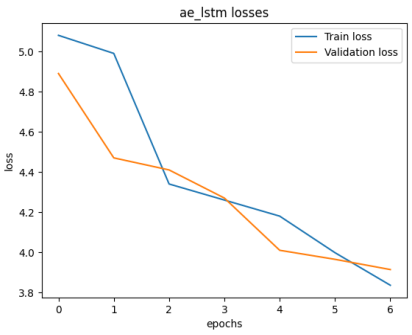
significant amount of time for complete processing through a model. Consequently, a decision was made to limit the vocabulary to only 10,000 tokens, with the condition that a word must appear in the corpus at least twice to be included.

Moreover, some models required token-by-token generation and the provision of target texts with masks, but some did not. These issues were successfully addressed by implementing a specialized branched training loop to accommodate all model types.

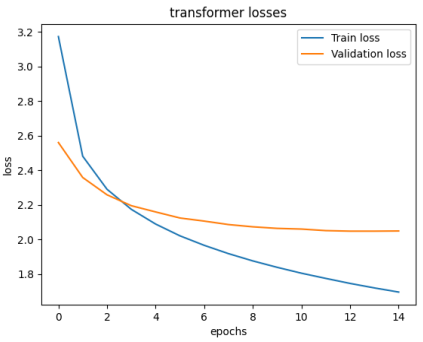
Due to the extended training times, it was not feasible to train models for a high number of epochs. As a result, I trained the LSTM for 10 epochs (given that it began to overfit after the third epoch) over 4 hours, the AutoEncoder LSTM for 7 epochs over 12 hours (due to the slow training nature of LSTMs), and the Transformer for 15 epochs over 7 hours. Training was done in Kaggle notebook with the batch size of 64 and on T4 GPU device. Unfortunately, the AutoEncoder LSTM notebook was not saved due to exceeding time limits, so I can only show 5 training epochs from my previous training attempt, only the loss plot was restored.



(figure 6)



(figure 7)



(figure 8)

Evaluation

For the evaluation, I've chosen metrics commonly used in machine translation tasks. The metrics I utilized are as follows:

Bilingual Evaluation Understudy (BLEU): This metric scores a machine-generated text by comparing it to one or more reference translations. It does this by examining the precision of matched clipped n-grams.

Recall-Oriented Understudy for Gisting Evaluation (ROUGE): ROUGE assesses a machine-generated text by considering precision and recall for matching unigrams and bigrams. In my case, I specifically used F1 scores for ROUGE1 and ROUGE2.

Translation Edit Rate (TER): TER measures the number of edits needed to improve a translation and make it both fluent and correct.

I performed this evaluation using a subset of ParaNMT, which consists of 1000 text pairs for assessing each model.

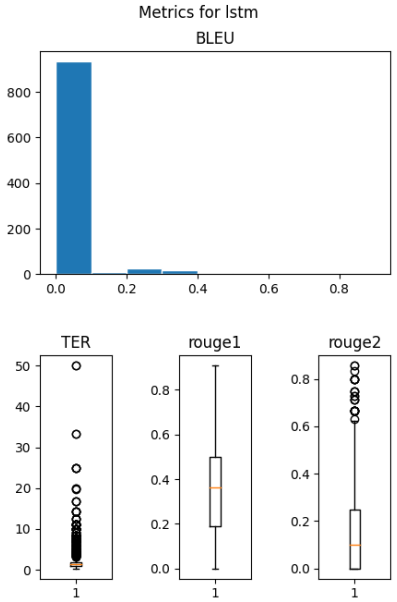
Results

The results of evaluation using metrics are far from perfect. However overall detoxification is quite good in case of transformers models. For example:
Input text: you are very stupid.
LSTM: you 're very bad . <eos>
AE LSTM: <unk> you 're too young .
Transformer: you 're very unreasonable .
T5: You're very naive.

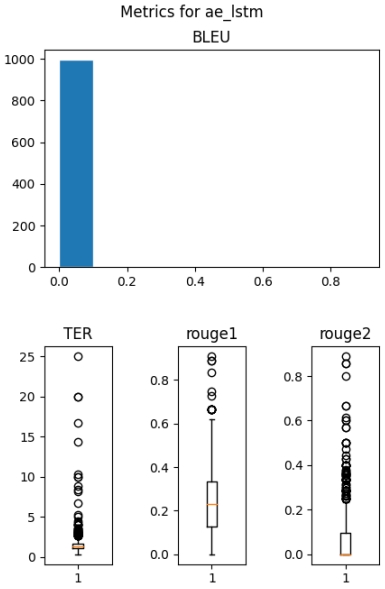
Metric evaluation:

Model/Metric	Average BLEU	Average ROUGE1 F1	Average ROUGE2 F1	Average TER
LSTM	0.02123	0.339749	0.154285	2.4416
AE LSTM	0.00249	1.5939	0.23202	0.0644
Transformer	0.066942	0.54086	0.30527	1.08759
T5	0.08652	0.56947	0.34273	0.64247

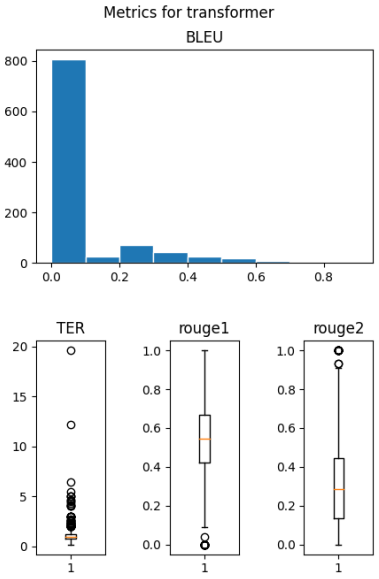
Box plots:



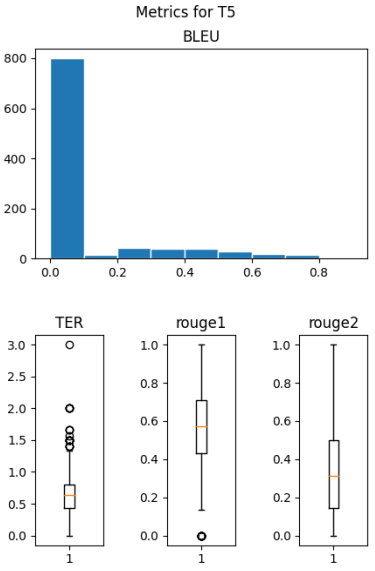
(figure 9)



(figure 10)



(figure 11)



(figure 12)

According to evaluation results the best model by metrics and generated text is T5-base pre-trained on ParaNMT.