Solution Building Report

Evaluation Metrics

Before diving into the implementation of the solution, I started by asking a simple question: what qualifies as a good solution?

I identified two criteria to evaluate the solution: the toxicity level of the predicted sentences, the semantic similarity between the predicted and target sentences, and the fluency of the predicted sentences.

For semantic similarity, popular metrics such as BLEU and ROUGE are available. However, as BLEU is more suitable for translation tasks and ROUGE is more suitable for summarization tasks, it seemed more sensible to use the cosine similarity between the embedding vectors of the predicted and target outputs. The more semantically similar our output is to the target sentence, the closer their embedding vectors will be.

For the toxicity level, I used an evaluator from Hugging Face with a pre-trained Roberta for toxic comment classification to assign a toxicity score to a sentence. The higher the score, the more toxic the comment is. To avoid misleading scores, sentences were split by punctuation and the toxicity score of each part was computed individually. The final score is the highest of these scores. Here's why: a sentence like "They all laughing at us, we will kick their ass" would give us a toxicity score of 0.003. However, if we split by punctuation, it would give us [0.001, 0.0004, 0.0007, 0.99, 0.0008], and choosing the highest score better represents the toxicity level of the sentence.

Solution Build

I will go over possible approaches I thought about that later led me to the final solution in sequential manner

Direct Dictionary Substitute

The first naive approach is to brute force word by word, and if we found a curse word, substitute by its equivalent from the dictionary.

However I didn’t go with that solution since it won’t be able to capture context of the sentence, and there might be a case where we need to rephrase the sentence not just blindly substitute

Mask Generation

To overcome the problem of the first approach of overlooking the context of the sentence. We could mask every curse word in a sentence, and finetune BERT to generate the masked word. In this way it will be context aware, However, the problem of not only generating a substitute of text and not rephrasing still persists.

Seq2Seq Text Generation

Generating the whole text seems like the logical approach to avoid two previously mentioned problems. but now the question is which model/architecture to use?

At first LSTMs and GRUs come to mind, but due to the complexity of the problem at hand, it will be really hard to train a model from scratch and probably will underfit,. Thus seems more logical to choose a pretrained model.

Encoder Decoder Model

I thought of finetuning encoder decoder model for text generation, in which both encoder and decoder are bert, bert2bert. However, the bottle neck layer will be randomly initialized and when I tried to finetune this (and training the bottle neck layer from scratch) it took too much resources without any results.

Final Solution: Finetune Alpaca

Then I thought of a stronger model than bert, in which I could train a very small subset of its parameters in an efficient way suitable for colab. I will elaborate more in the final solution report.

Dataset Analysis and Cleaning

Not to confuse the model I picked subset that has ≥0.9 toxicity level in the reference sentences and less than 0.2 in translation sentences. Furthermore, I removed any redundant punctuation