Final Solution Report

Introduction

This assignment is an attempt to solve the Text Detoxification Problem.

Text detoxification can be defined as transforming text with high level of toxicity - that is, containing rude, disrespectful, or inappropriate words or phrases, that is likely to make another people leave a conversation - into the text with the same meaning, but with non-toxic style by removing or replacing the toxic words.

My approach to this problem is to consider toxic and non-toxic sentences as different languages and to create a “translator” from the toxic language to the non-toxic language.

Data Analysis

In the given dataset there are 4 columns that are useful for this translation task: ‘reference’, with a sentence in toxic language, ‘translation’, its translated version in non-toxic language, ‘ref_tox’ and ‘trn_tox’ with numerical values of toxicity in ‘reference’ and ‘translation’ columns correspondingly.

It is worth noticing that some sentences in the given dataset are in wrong columns: some sentences in the toxic language are in the ‘translation’ column with their equivalents in the ‘reference’ column. This can be easily detected and corrected by comparing the corresponding values in ‘ref_tox’ and ‘trn_tox’ columns that are also misplaced in this case.

In overall, the sentences are the usual string of text taken from ParaNMT-50M corpus, which is a dataset for training paraphrasing sentence embeddings, so it should be enough to train such a “translation” model quite well.

Model Specification & Training Process

As a preprocessing step I only make sentences lowercase and split the punctuation from words. Due to the fact the result of translation must be an intelligible sentence, I did not utilize other standard NLP techniques such as stop-words removal, stemming, and lemmatization. Then, I used two dictionaries that map each lowercased word and punctuation mark to an integer and back.

As a main module for text generation I use LSTM. Experiments have shown that using more than one layer for LSTM shows worse results, so I decided to use only one layer for it. Additionally, I use one fully connected layer after the LSTM.

As an additional improvement for training this NN, I decided to use two different loss functions - if padding was correct, then the loss would be calculated only on the meaningful part of the sentence, and if the padding was incorrect, then the loss would be calculated on the whole sentence. With this addition, computation power was mostly spent on learning the language instead of padding.

Unfortunately, the training process was stopped because I ran out of GPU quota, so the current version of the model shows bad results.

Results

I did not utilize any metric for evaluating the model's performance because it is a very subjective metric; also, using the dataset is not the most reliable way.

Anyways, the existing model shows exceptionally bad results. Here are some of them:

Input: shut up
Output: be quiet

Input: I like that shit.
Output: wicked like that that .

Input: Hey, leave the poor bastard alone!
Output: wicked , , the poor the .