Final Solution Report

By Vladislav Urzhumov, BS21-AI-01 (v.urzhumov@innopolis.university)

Introduction

To start with, the project dedicated to the present github is a solution for the **Text Detoxification** task, raised in many researches and in a media. Toxic contexts and bad words flood the internet and, particularly, social media. What if we add a detoxification filter to each and every message, post and article posted in the internet? World would be a better place.

However, manual detoxification or algorithm-based detoxification are either resource-consuming or inefficient, or both. Thus, we need a better solution, which embraces the power of Artificial Intelligence. Large Sequence-to-Sequence models seem to be a good choice for the kick start of this journey.

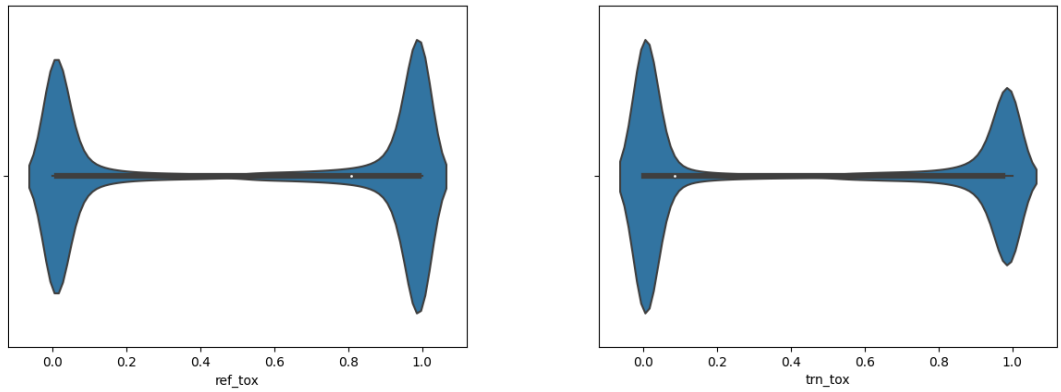
Dataset

Here, the *paraNMT* dataset is used. It consists of pairs of text sequences; each pair has corresponding toxicity values of both variants. Toxicity is measured from 0 to 1, the less – the better. Toxic sentences come in pairs with neutral ones, although having the same meaning preserved. Moreover, dataset has additional information, such as pair similarity and length difference.

Data exploration

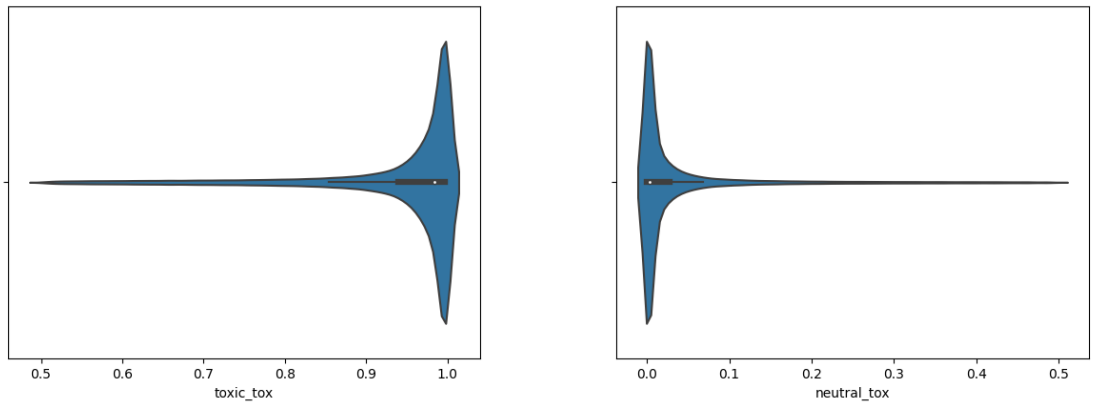
Dataset exploration and visualization will simply show us the data distribution, challenges and erroneous data. We will use python scripts along with *pandas, seaborn* and *matplotlib* libraries to explore and visualize.

First thing to note, pairs are shuffled. Meaning neither “reference” nor “translation” column have all their values on one side of toxicity. We can visualize this problem using the violin plots of “ref_tox” and “trn_tox”:



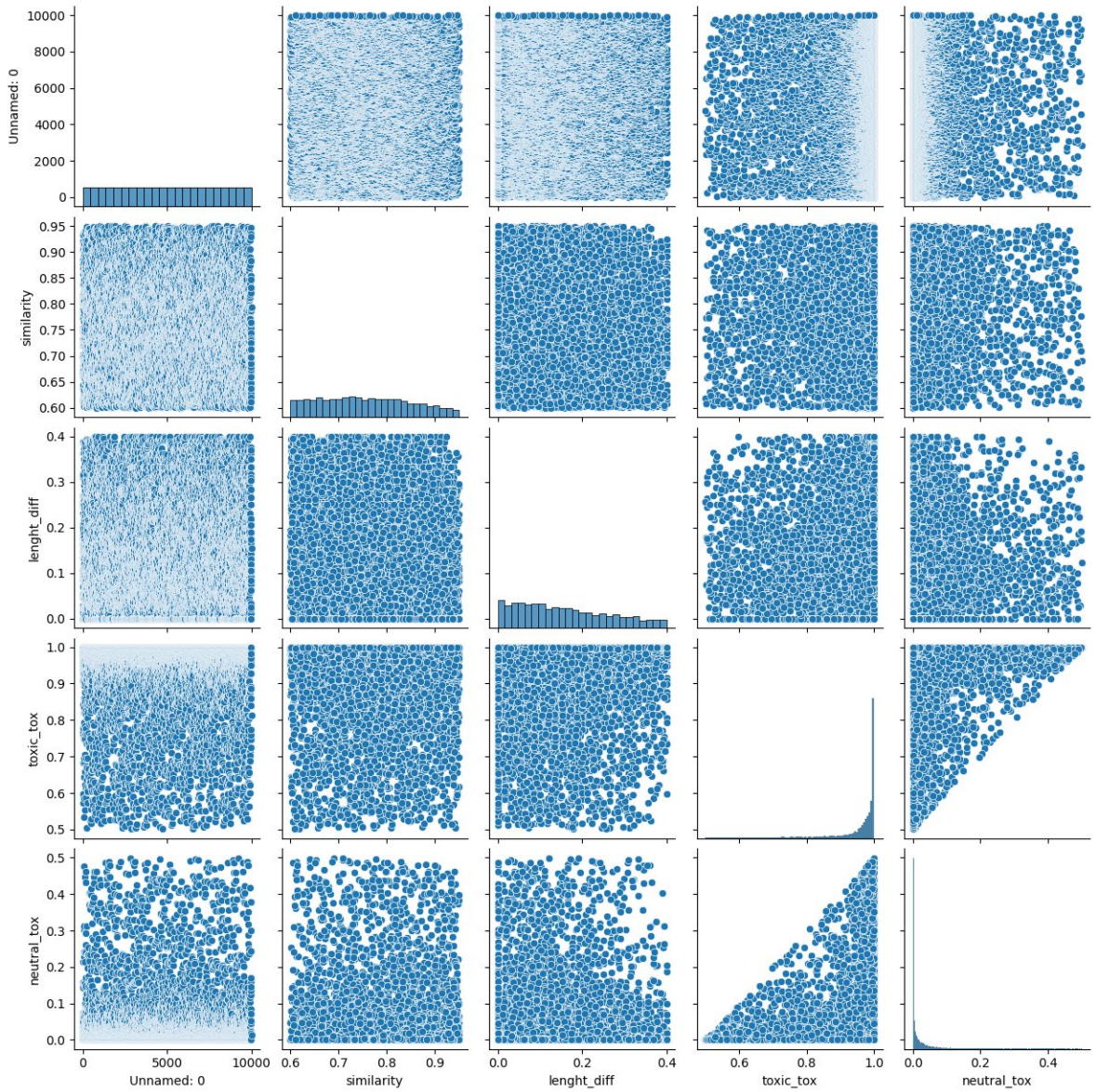
Thus, one has to determine toxic and neutral manually by their toxicity value as a preprocessing step and create two distinguished columns. Let us name those columns “toxic” and “neutral” correspondingly. The condition of such a division must be set as “the toxicity value of instance in

the neutral column must be less than or equal to the toxicity value of instance in the toxic column”. Let us visualize the results of such preprocessing:



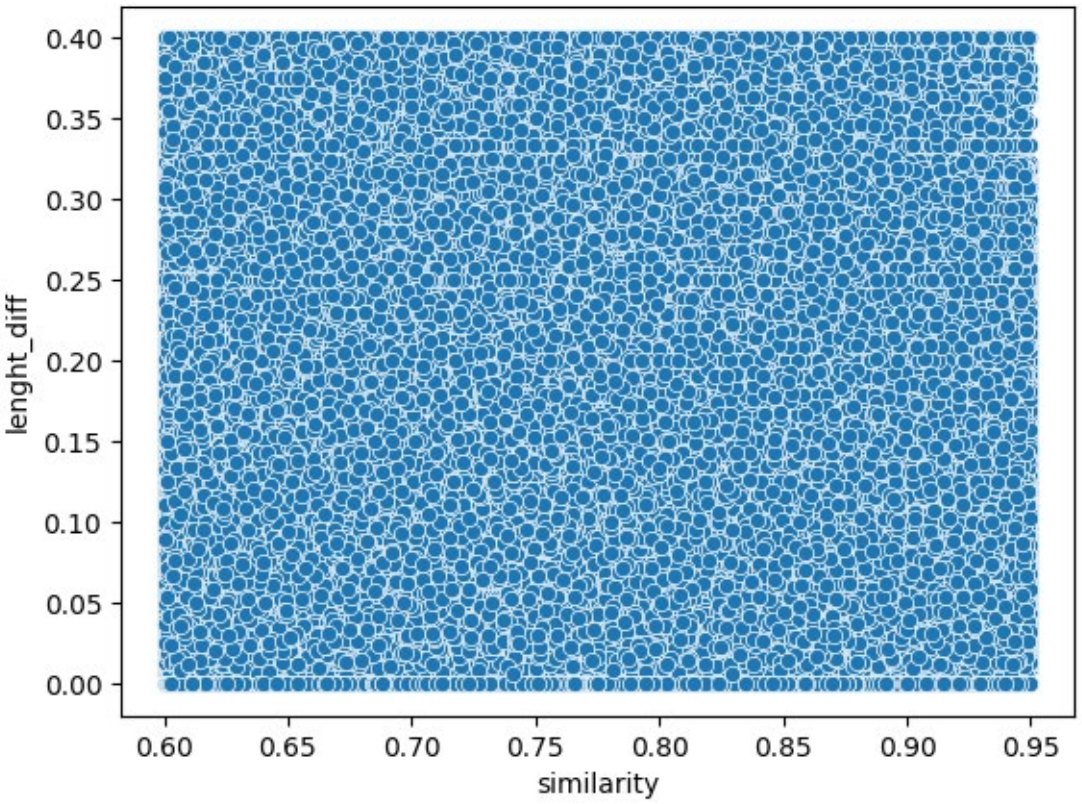
Values of toxicity in the “toxic” column are in range (0.5, 1), whereas values of toxicity in the “neutral” column are in range (0, 0.5). Condition is satisfied.

Secondly, the data distribution is even:



Thus, we can quite safely crop the dataset for the sake of time economy. Dataset is quite large (more than 500’000 pairs) for our purpose, since we want to fine-tune a pre-trained model.

Thirdly, let us look on the length difference and similarity of the pair. To be more precise, on their mutual dependency. We will use scatter plot:



Fairly enough, there is no direct correlation of these two columns. Sentences may differ in size, however it does not affect their semantic similarity (meaning).

Model selection

Once we finished with the dataset exploration, we can move to the model selection. Different models were tested in this project, starting from the ones mentioned in the reference paper and its authors’ works (these models are T5, GPT and Bert), finishing by specific models, such as Prophet Net by Microsoft.

Worth mentioning that the lack of resources and computational performance were a huge issue and stood a solid place as a criterion for the model choice.

As a final solution, Prophet Net model, which was trained on Google Colab’s resources, showed the best performance during the project lifetime, thus it is referred to as the “best” model here and later in the report.

Training

The training was made as simple as possible. The efficient trainers from the huggingface’s library named “transformers” were used to embrace the power of CUDA and reduce computational expenses. Challenges and hypothesis tests are described in the “Solution building

report". Moreover, code is well-structured and comment-covered with docstrings, thus one may check it.

To be concise, just a few notes about hyperparameters and configurations:

- "best" model is trained using model-specific Seq2SeqTrainer from transformers
- Batch collate is performed via transformers library's DataCollatorForSeq2Seq
- The metric used is sacrebleu
- Learning rate is dynamic. "best" model training started with learning_rate=2e-5
- Batch size is adjusted so CUDA never runs out of memory (Google Colab endured 32, locally project's author had to stick with 4)
- Dataset was cropped to 10000 in train and 2000 in both validation and test
- Number of epochs was 10

Prediction

Any user of current project solution has the possibility to predict the detoxificated paraphrased version of his/her sentence. For this, they may just use the "predict" function from src/models/predict_model.py while having some trained model in the models/best directory. After the training, if flag was not set to False, the model is automatically saved as "best" to the abovementioned directory.

Examples of the user text detoxification can be found in the jupyter notebook 2.0 ("2.0-prophet-model.ipynb") or in main.py. However, one may want to see some textual representation example here in the report. Here is one:

```
Initial message: Who the hell is ringing my bell?,
Paraphrased: who's ringing the doorbell at my house?
```

Evaluation

Evaluation is made on the validation part of the dataset and is measured also as loss and sacrebleu metric. The training of T5 model showed the progress in those two, since the validation loss was going down along with train loss: it went down by 15% during those 10 epochs of training.

Prophet Net showed good overall performance and is referred as the best model, however it's validation loss stopped decreasing mid-training, while train loss was still going down. Thus, we faced either stagnation or overfitting on the validation part of the dataset.

Conclusion

Any solution which removes bad words and paraphrases the toxic statements fairly well is a positive aid for any internet platforms. Fine-tuned Sequence-to-Sequence models showed an excellent performance in the detoxification task. However, some challenges were faced due to the raw dataset problems and, more significantly, lack of computational resources. These challenges were overcome and now we have a solution and well-structured code along with visuals and reports. Enjoy!

Thanks for reading, have a nice day! Feel free to contact in case of any questions.