# Text Detoxification

Almaz Dautov
Innopolis University

## 1.Dataset

### 1.1 Main dataset consists of two datasets ParaNMT filtered and Paradetox.

> ParaNMT filtered
> The dataset is a subset of the ParaNMT corpus (50M sentence pairs). The filtered ParaNMT-detox corpus (500K sentence pairs)
>
> ParaDetox: Detoxification with Parallel Data (English)
> This repository contains information about Paradetox dataset -- the first parallel corpus for the detoxification task -- as well as models and evaluation methodology for the detoxification of English texts. The original paper "ParaDetox: Detoxification with Parallel Data" was presented at ACL 2022 main conference.

### 1.2 Dataset for baseline classifier.

> Positive and Negative Words GitHub.
> Bad Words dataset: Kaggle
> Inappropriate Words dataset Kaggle

## 2.Data analysis

For the primary dataset, we retain only the references where the translated toxicity score is lower than that of the references.

Regarding the baseline classifier dataset, we follow a specific approach. Words can often possess both negative and positive meanings depending on the context. In our categorization, we label them as "Negative" to minimize the occurrence of False Positive identifications. While this approach can improve precision scores, it may also result in a potential decrease in recall, potentially leading to more instances of False Negatives. However, for our particular task, this trade-off isn't critically significant, as we have the flexibility to substitute positive words with suitable alternatives when necessary.

## 3.Metrics

Metric for evaluation of the model's performance is the J metric, which is the multiplication of sentence-level style accuracy, content preservation, and fluency. Style accuracy (ACC) is measured with a pre-trained toxicity classifier. Content preservation (SIM) is evaluated as the similarity of sentence-level embeddings of the original and transformed texts computed by the model of Wieting et al. (2019). Fluency (FL) measured

with the classifier of linguistic acceptability trained on the CoLA dataset (Warstadt et al., 2019). J is computed as the average of their sentence-level product. [1]

### 3.1 Transfer accuracy (ACC)

Given an output sentence ¯sj and a target style j, a common way of measuring transfer success is to train a classifier to identify the style of a transferred sentence and report its accuracy ACC on generated sentences (i.e., whether ¯sj has a predicted style of j). 14 of 23 surveyed papers implement this style classifier with a 1-layer CNN (Kim, 2014). However, recent large Transformers like BERT (Devlin et al., 2019) significantly outperform CNNs on most NLP tasks, including style classification. Thus, we build our style classifier by fine-tuning RoBERTa-large (Liu et al., 2019) on all our datasets, leading to significantly more reliable ACC evaluation.7

### 3.2 Semantic similarity (SIM)

A style transfer system can achieve high ACC scores without maintaining the semantics of the input sentence, which also motivates measuring how much a transferred sentence deviates in meaning from the input. 15 / 23 surveyed papers use n-gram metrics like BLEU (Papineni et al., 2002) against reference sentences, often along with self-BLEU with the input, to evaluate semantic similarity. Using BLEU in this way has many problems, including (1) unreliable correlations between n-gram overlap and human evaluations of semantic similarity (Callison-Burch et al., 2006), (2) discouraging output diversity (Wieting et al., 2019), and (3) not upweighting important semantic words over other words (Wieting et al., 2019; Wang et al., 2020). These issues motivate us to measure semantic similarity using the subword embedding-based SIM model of Wieting et al. (2019), which performs well on semantic textual similarity (STS) benchmarks in SemEval workshops (Agirre et al., 2016).

### 3.3 Fluency (FL)

A system that produces ungrammatical outputs can still achieve high scores on both ACC and SIM, motivating a separate measure for fluency. Only 10 out of 23 surveyed papers did a fluency evaluation; 9 of which used language model perplexity, which is a poor measure because (1) it is unbounded and (2) unnatural sentences with common words tend to have low perplexity (Mir et al., 2019; Pang, 2019). To tackle this we replace perplexity with the accuracy of a RoBERTa-large classifier trained on the CoLA corpus (Warstadt et al., 2019), which contains sentences paired with grammatical acceptability judgments. In Table 1, we show that our classifier marks most reference sentences as fluent, confirming its validity. [2]

$$\text{J(ACC, SIM, FL)} = \sum \frac{ACC(X) * SIM(X) * FL(X)}{|X|}$$

## 4.Baseline Hypothesis: Building a Toxic Word Classifier and substituting toxic words with synonyms from NLTK Lesk

### 4.1 Toxic Word Classifier

For our classifier, we will employ a straightforward logistic regression model. We will train this model to distinguish between toxic and non-toxic words. To accomplish this, we will use the following datasets for our baseline classifier:

4.1.1 Positive and Negative Words: We will source positive and negative words from GitHub.

4.1.2 Bad Words dataset: We will utilize the dataset available on Kaggle.

4.1.3 Inappropriate Words dataset: Additionally, we will leverage the dataset provided on Kaggle

### 4.2 Lesk Algorithm

Performs the classic Lesk algorithm for Word Sense Disambiguation (WSD) using the definitions of the ambiguous word. Given an ambiguous word and the context in which the word occurs, Lesk returns a Synset with the highest number of overlapping words between the context sentence and different definitions from each Synset.

### 4.3 Evaluation:

| ACC | SIM | FL | J | BLEU |
|---|---|---|---|---|
| 0.5798 | 0.5306 | 0.6789 | 0.2075 | 0.3868 |

## 5. Hypothesis 1: Pre-trained Toxic Word Classifier and pre-trained paraphraser

We can try to increase the metrics by enhancing both the paraphrasing model and the classifier.

### 5.1 Toxic Word Classifier

roberta-base pre-trained on **Jigsaw Unintended Bias in Toxicity Classification**

| Challenge | Year | Goal | Original Data Source | Detoxify Model Name | Top Kaggle Leaderboard Score | Detoxify Score |
|---|---|---|---|---|---|---|
| Jigsaw Unintended Bias in Toxicity Classification | 2019 | build a model that recognizes toxicity and minimizes this type of unintended bias with respect to mentions of identities. You'll be using a dataset labeled for identity mentions and optimizing a metric designed to measure unintended bias. | Civil Comments | `unbiased` | 0.94734 | 0.93639 |

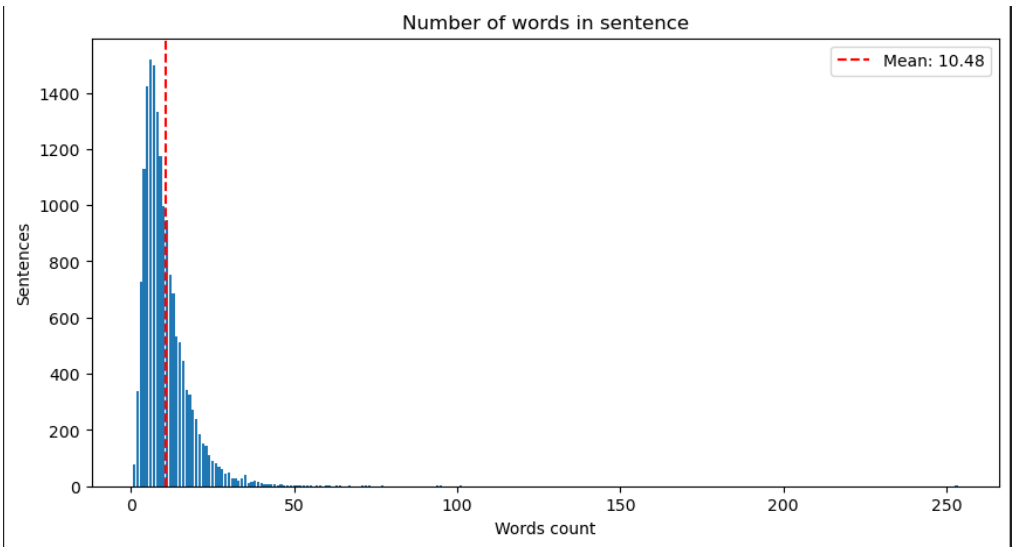
### 5.2 Paraphrasing model

PEGASUS or Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence models, fine-tuned for paraphrasing. It uses self-supervised objective Gap Sentences Generation (GSG) to train a transformer encoder-decoder model.

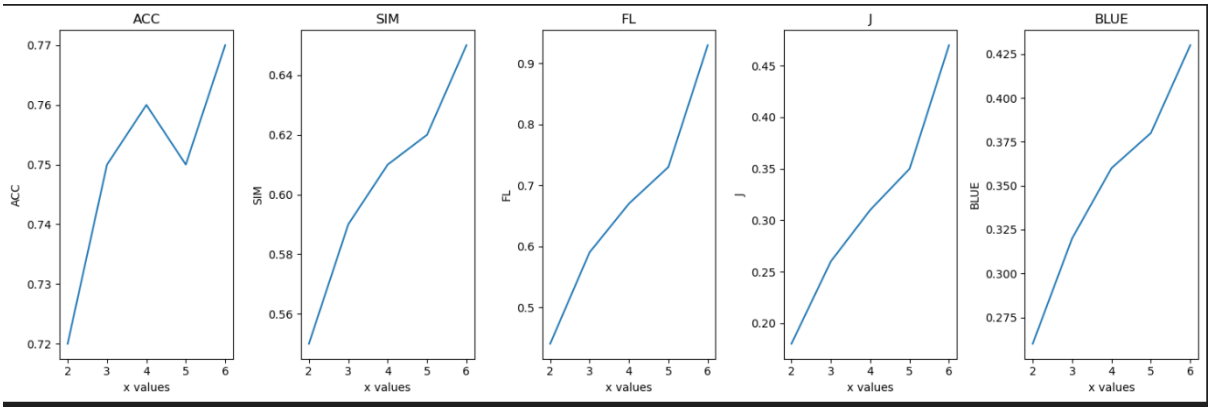
### 5.3 n word

Previously, we conducted word classification and paraphrasing on a per-word basis within a sentence. Now, with the Seq2Seq model, we have the capability to paraphrase multiple words in a sentence simultaneously. To facilitate this, we introduce a hyperparameter, 'n', which signifies the number of words considered in a single token. This feature is not available in nltk.lesk, which exclusively operates on individual words and not entire sentences.

Given the model's computational demands during inference, we will work with a smaller subset of our test data, consisting of 1000 samples. We will perform a grid search on the 'n' hyperparameter to optimize its value, as this can significantly impact model performance.




Given that the average sentence length is approximately 10 words, let's conduct a gridsearch for the hyperparameter 'n.' We'll explore values of 'n' up to half the sentence length and Additionally, we consider the entire sentence context.




As shown in the plot, when 'n' increases, our metrics also rise. Our key metric, 'J,' encompasses all others and demonstrates consistent improvement as 'n' grows. Therefore, our future paraphrasing approach will involve using complete sentences, without dividing them into 'n' words.

## 5.4 Evaluation:

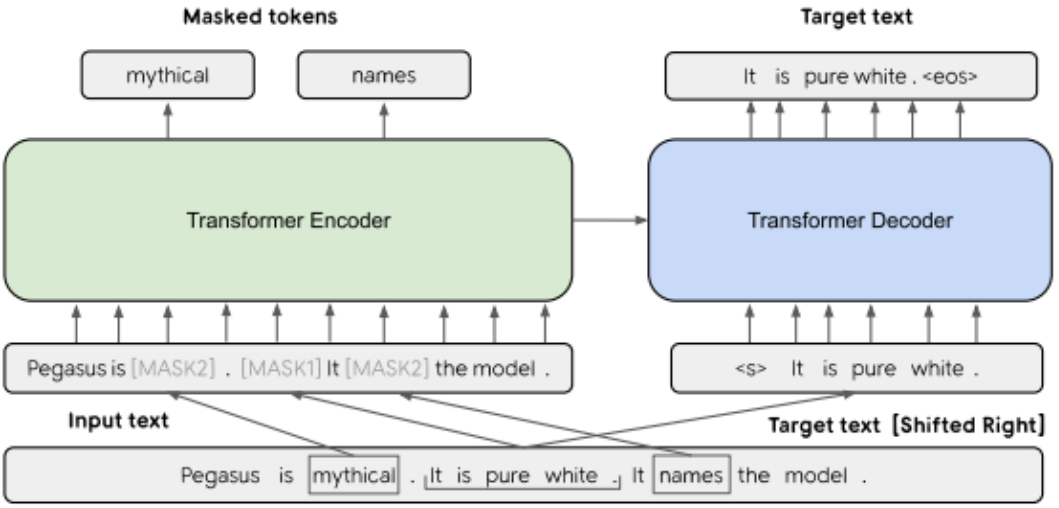
| ACC | SIM | FL | J | BLEU |
|---|---|---|---|---|
| 0.7569 | 0.6511 | 0.9333 | 0.4640 | 0.4248 |

## 6. Hypothesis 2: Fine-tune paraphraser on detoxifying dataset

Now, we've come to realize that the most effective approach for text detoxification is to utilize the entire sentence within transformer models. Let's proceed with fine-tuning the Pegasus model for this purpose. In doing so, we won't require a separate classifier anymore, as our model will simultaneously classify and paraphrase text.

### 6.1 Model

The base architecture of PEGASUS is a standard Transformer encoder-decoder.In PEGASUS, the model generates important sentences that have been removed or masked from an input document as a unified output sequence. This process resembles the extraction of key content, similar to an extractive summary. [3]




### 6.2 Training

The main dataset was divided into three subsets for training, validation, and testing. A small portion, 5%, was allocated for testing, while the remaining 95% was used for training and validation. Within the training set, 20% was set aside for validation. The model was trained with a learning rate of 2e-5, incorporating a weight decay of 0.01, over the course of 5 training epochs.

| Epoch | Training Loss | Validation Loss | Acc | Sim | Fl | J | Blue |
|---|---|---|---|---|---|---|---|
| 1 | 0.392100 | 0.357948 | 0.714656 | 0.750274 | 0.906254 | 0.494035 | 0.554488 |
| 2 | 0.365300 | 0.344334 | 0.748804 | 0.754590 | 0.908101 | 0.520794 | 0.563554 |
| 3 | 0.354200 | 0.337801 | 0.761951 | 0.755998 | 0.909038 | 0.531385 | 0.566634 |
| 4 | 0.345600 | 0.335123 | 0.768151 | 0.757607 | 0.909578 | 0.536959 | 0.569053 |
| 5 | 0.343400 | 0.334158 | 0.772030 | 0.757325 | 0.909855 | 0.539474 | 0.569136 |

### 6.3 Evaluation:

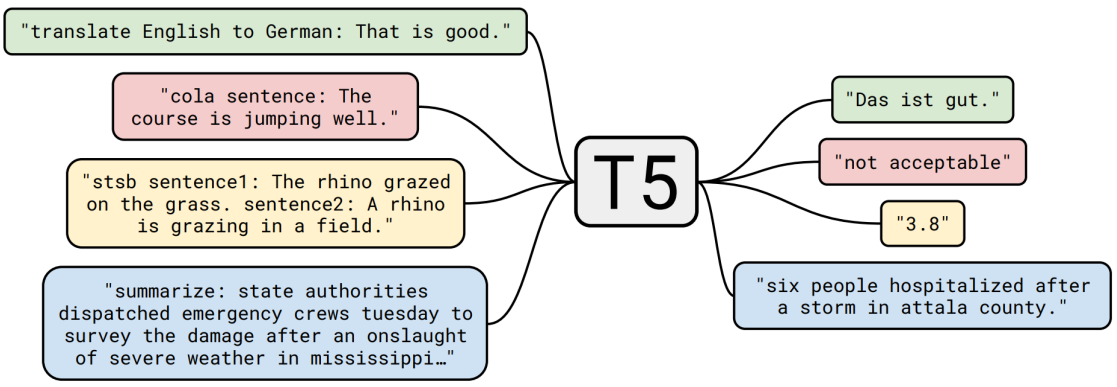
| ACC | SIM | FL | J | BLEU |
|---|---|---|---|---|
| 0.7727 | 0.7600 | 0.9131 | 0.5421 | 0.5710 |

## 7. Hypothesis 3: Fine-tune t5-small on detoxifying dataset

Let's explore the performance of another transformer with a reduced number of parameters. T5-Small offers a balance between model size and performance, making it a practical choice for text detoxification.

### 7.1 Model

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task, e.g., for translation: translate English to German: …, for summarization.




### 7.2 Training

The training parameters remain consistent with those of PEGASUS.

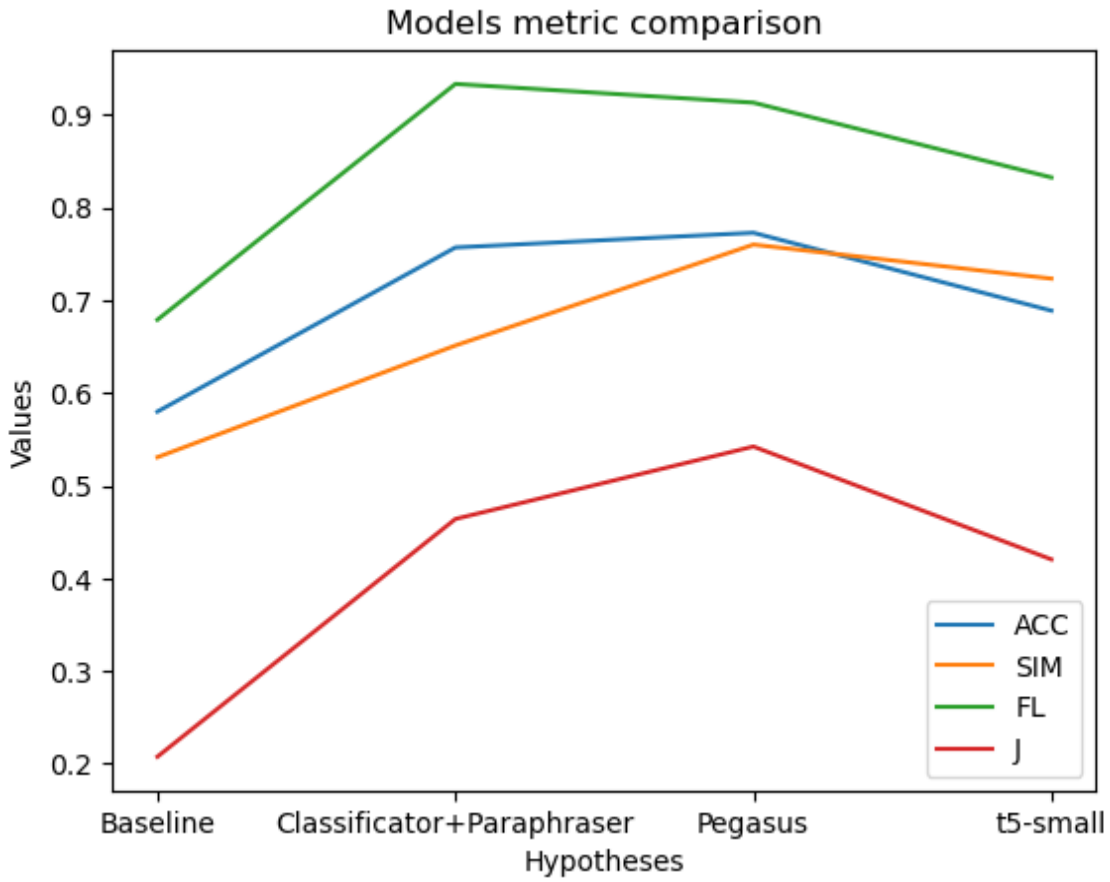
| Epoch | Training Loss | Validation Loss | Acc | Sim | Fl | J | Blue |
|---|---|---|---|---|---|---|---|
| 1 | 0.227700 | 0.202000 | 0.626425 | 0.706339 | 0.822377 | 0.369215 | 0.499427 |
| 2 | 0.214800 | 0.193629 | 0.658840 | 0.715029 | 0.829025 | 0.397430 | 0.509289 |
| 3 | 0.208400 | 0.189763 | 0.678553 | 0.718720 | 0.831062 | 0.412158 | 0.514119 |
| 4 | 0.205900 | 0.187898 | 0.685357 | 0.720615 | 0.831987 | 0.418187 | 0.516174 |
| 5 | 0.204300 | 0.187309 | 0.689156 | 0.720980 | 0.832544 | 0.420949 | 0.516778 |

7.3 Evaluation

| ACC | SIM | FL | J | BLEU |
|---|---|---|---|---|
| 0.6887 | 0.7233 | 0.8324 | 0.4203 | 0.5192 |

## 8. Comparison

| **Original** | **baseline** | **paraphraser+ classifier** | **pegasus** | **t5-small** |
|---|---|---|---|---|
| i gotta have her i fell n shit | i obtain Ta have her i hide N take a crap | I had to have her. | I have to have her. | I have to have her i fell n shit |
| I just want a glass of water. Oh, shit! | i just want a glass of water supply Buckeye State take a crap | I just want a drink of water. | I just want a glass of water. | I just want a glass of water. |
| We have a big fuckin' problem. | we have a large fuckin problem | We have a big problem. | we have a big problem. | we have a big problem |
| Some freak just sent me this. | some lusus naturae just send me this | Someone just sent me this. | someone sent me this. | a freak sent me this. |




## 9. References

[1] Text Detoxification using Large Pre-trained Neural Models, David Dale et. al.
[2] Reformulating Unsupervised Style Transfer as Paraphrase Generation, Kalpesh Krishna et. al.
[3] PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization, Jingqing Zhang et. al