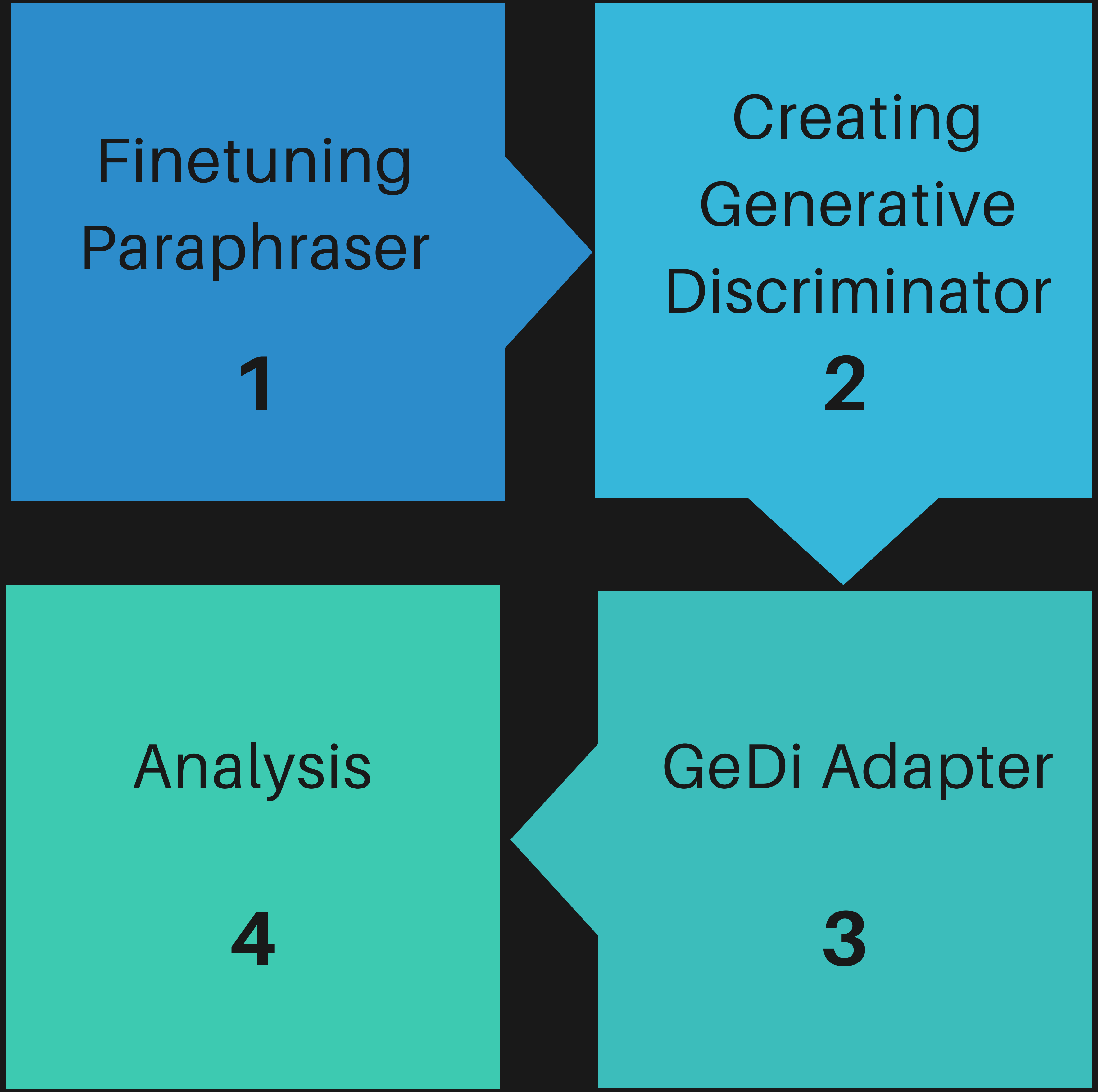
Text Detoxification using Large Pre-trained Neural Models

Team:
Nanditha Merugu (2020102061)
Anantha Lakshmi Yadavalli(2020101103)
Harshavardhan Thatipamula(2020101106)

PROJECT CYCLE



Overview

- Rewriting the text to: preserver the text meaning, eliminate the toxic style.
- Example:

They are all communists who hate the USA

$\downarrow$

They are all communists who dislike the USA

Understanding the problems

Problem 1

Despite the aim of maintaining the original content, the process of altering the sentence's style often leads to substantial changes in its meaning.

Solution

Detoxification requires better preservation of the original meaning than other style transfer tasks, so it should be done differently.

Problem 2

There are TST(Text Style transfer) models. Both content and style are ambiguous concepts. TST often changes the original meaning.

Methodology

Our method combines two recent ideas:

Two novel methods for text detoxification:
(a) ParaGedi (paraphrasing GeDi)
(b) CondBERT (conditional BERT)

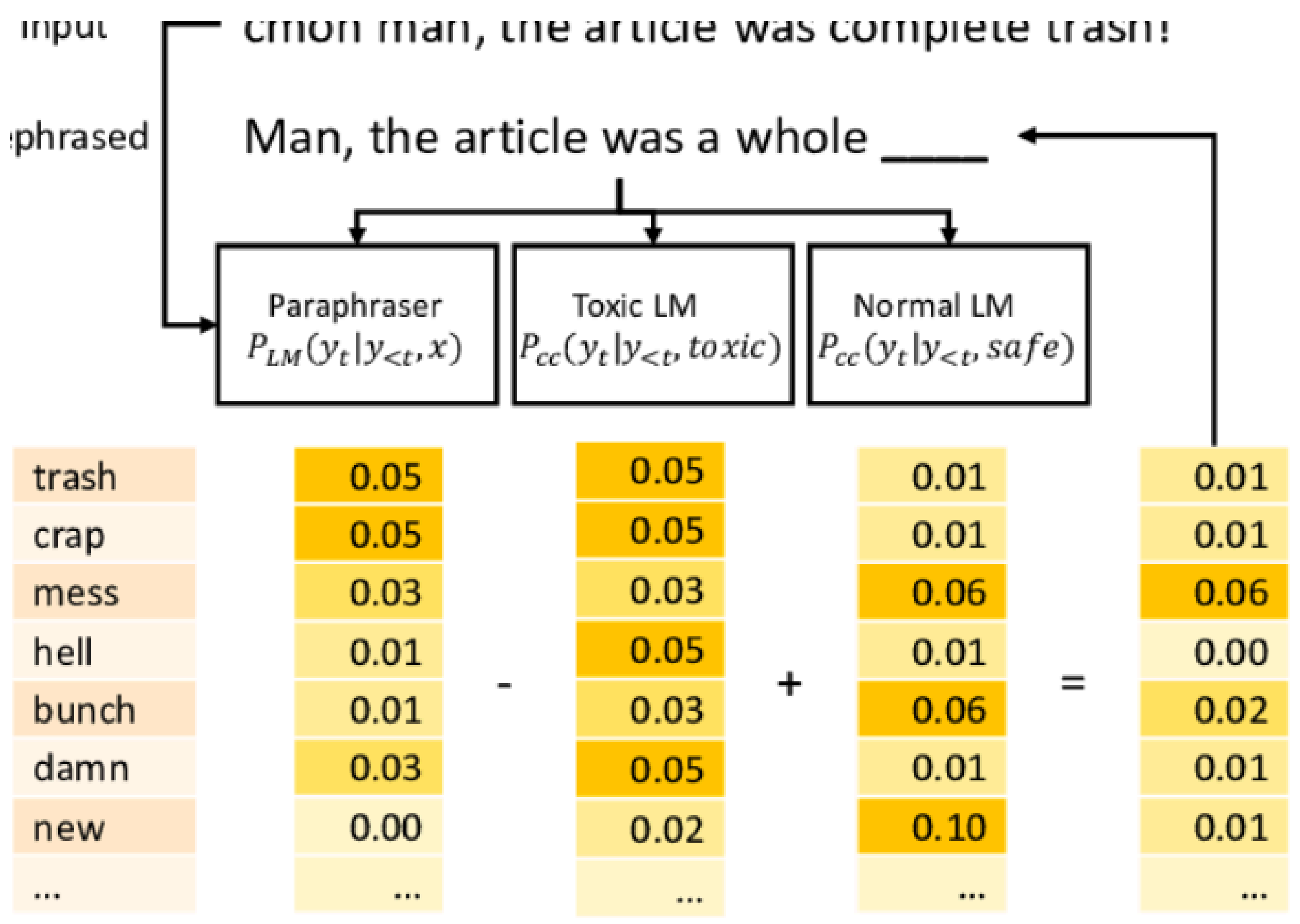
- A parallel toxic/safe corpus retrived from the ParaNMT dataset.

ParaGeDi

The method is based on two ideas:

1. A large pre-trained paraphraser model can preserve the meaning.
2. A pre-trained language model conditional on style can control the style.

- Then the two models can be combined using the Bayes rule.


input
cmon man, the article was complete trash!
phrased
Man, the article was a whole ____
Paraphraser $P_{LM}(y_t|y_{<t},x)$
Toxic LM $P_{cc}(y_t|y_{<t},toxic)$
Normal LM $P_{cc}(y_t|y_{<t},safe)$
-
+
=

trash	0.05	0.05	0.01	0.01
crap	0.05	0.05	0.01	0.01
mess	0.03	0.03	0.06	0.06
hell	0.01	0.05	0.01	0.00
bunch	0.01	0.03	0.06	0.02
damn	0.03	0.05	0.01	0.01
new	0.00	0.02	0.10	0.01
...	...	...	...	...

Formulas used

$$P(y_t|y_{<t},x,c) \propto P_{LM}(y_t|y_{<t},x)P(c|y_t,y_{<t},x)$$
$$\approx P_{LM}(y_t|y_{<t},x)P_D(c|y_t,y_{<t})$$

$$\mathcal{L}_G = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i}\log P(y_t^{(i)}|y_{<t}^{(i)},c^{(i)})$$

$$\mathcal{L}_D = -\frac{1}{N}\sum_{i=1}^{N}\log P(c^{(i)}|y_{1:T_i}^{(i)})$$

$$\mathcal{L}_{ParaGeDi} = \lambda\mathcal{L}_D + (1-\lambda)\mathcal{L}_G$$

By approximating the style transfer task as paraphrasing, we can separate the training requirements for the paraphrasing model (which needs parallel data) and the style model (which only needs text labeled for style, not necessarily in parallel). Additionally, we introduce an optional Reranker to refine the style of the generated text by assigning different weights to the hypotheses based on their adherence to the desired style.

Task 1:

Implementing and training of Generative Discriminator

The **gpt2-Medium** model served as the basis for our discriminator model (Gedi), which is designed for text generation. We utilized the renowned JigSaw dataset to train this model, first filtering out irrelevant data. A binary toxicity_classifier was trained to identify toxic comments within the dataset. These comments were then divided into shorter sentences. The model was trained by adding the classifier class codes (normal, toxic) to the beginning of the sentences and using class-conditional language modeling. The **gedi_training.py** file contains the implementation of this process.

Task 2:

Mining the dataset and Finetuning of the Paraphraser

The second step involved implementing the paraphrasing component of Para-Gedi. We began by fine-tuning the T5-base paraphrasing model on PAWS, MSRP, and Opinosis datasets. Subsequently, we further fine-tuned the model using a refined subset of the para-nmt dataset. The refinement process is outlined in **para-nmt-extraction.py**, while the fine-tuning of T5 is detailed in **finetuned-t5-on-extracted.py**. For comparison purposes, we also employed a paraphraser fine-tuned on a random sampling of the para-nmt dataset. We plan to conduct a more comprehensive evaluation to determine the effectiveness of the refined model compared to the randomly sampled counterpart.

Task 3:

Implementing the J metric for evaluation

- To assess the performance of our model relative to other baseline models, we implemented the J metric in the metric.py file.
- This metric measures the effectiveness of our model in achieving three objectives: **style transfer**, **content preservation**, and **fluency**.
- The J metric is calculated by multiplying the sentence-level scores for each of these objectives. Since these objectives are inversely correlated, a higher J metric indicates a better balance between them.

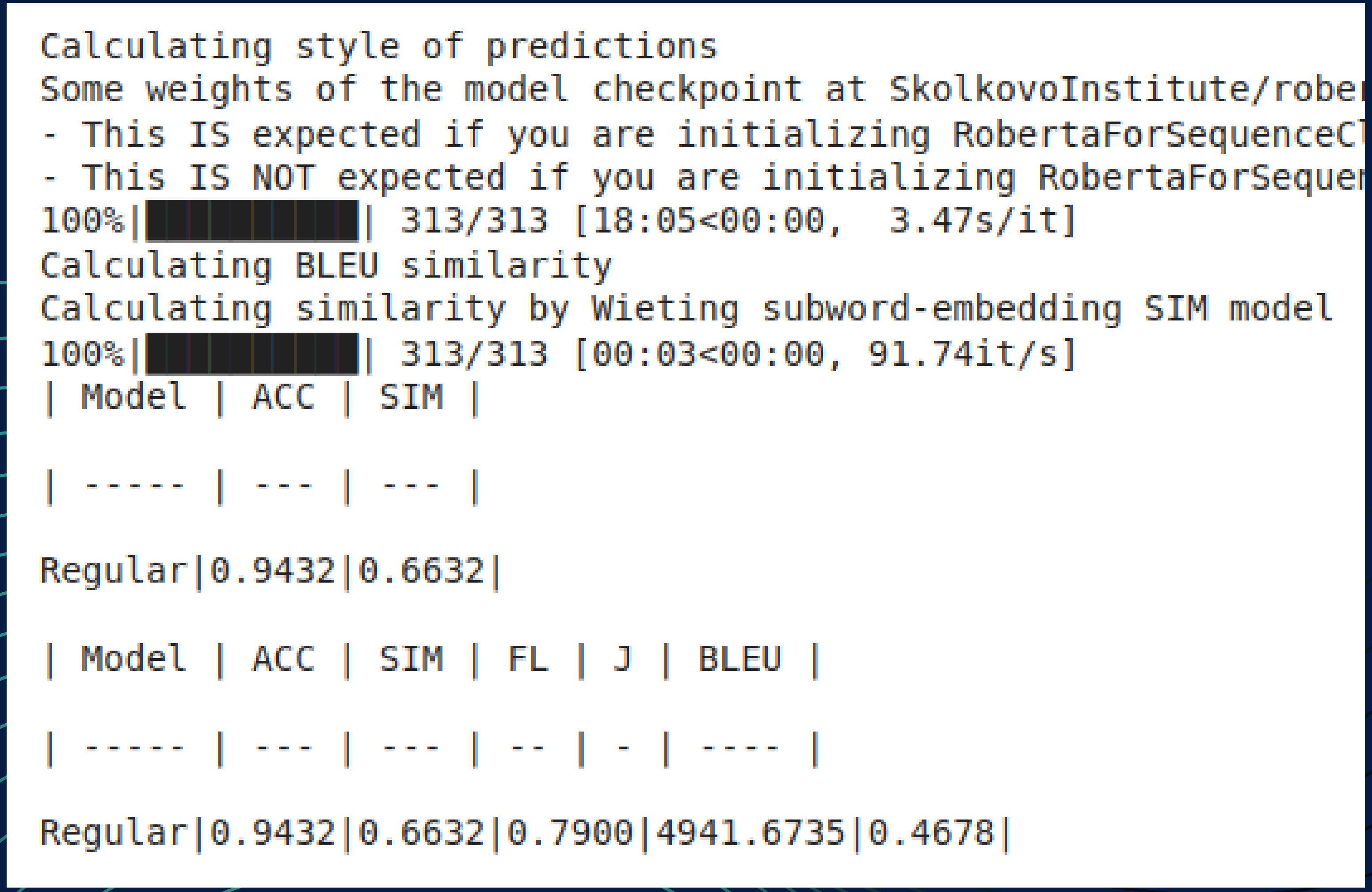
Task 3 cont.

Implementing the J metric for evaluation

- ACC: mean score of RoBERTA-based toxicity classifer
- SIM: is evaluated based on the similarity between sentence-level embeddings of the original and transformed texts. This is achieved by employing the trained SIMILE model.
- Fluency (FL): is measured using a classifier of linguistic acceptability that has been trained on the CoLA dataset.
- J : mean product of sentence level ACC, SIM and FL

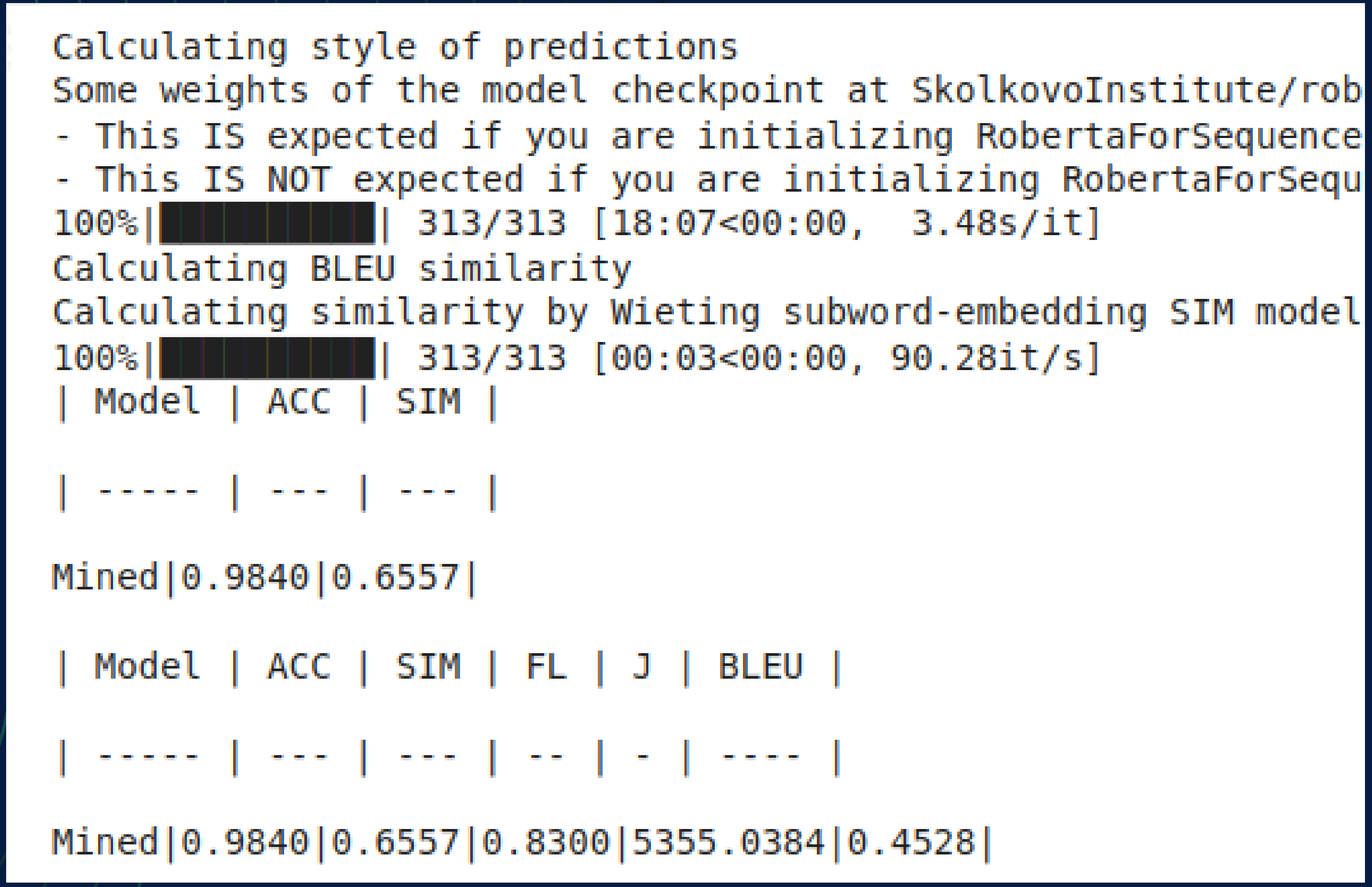
Results

For the **ParaGeDi**, we got the below scores in 2-decimal precision for Regular and MinedModel:-



```
Calculating style of predictions
Some weights of the model checkpoint at SkolkovoInstitute/robe
- This IS expected if you are initializing RobertaForSequenceCl
- This IS NOT expected if you are initializing RobertaForSequen
100%|██████████| 313/313 [18:05<00:00,  3.47s/it]
Calculating BLEU similarity
Calculating similarity by Wieting subword-embedding SIM model
100%|██████████| 313/313 [00:03<00:00, 91.74it/s]
| Model | ACC | SIM |

| ----- | --- | --- |

Regular|0.9432|0.6632|

| Model | ACC | SIM | FL | J | BLEU |

| ----- | --- | --- | -- | - | ---- |

Regular|0.9432|0.6632|0.7900|4941.6735|0.4678|
```

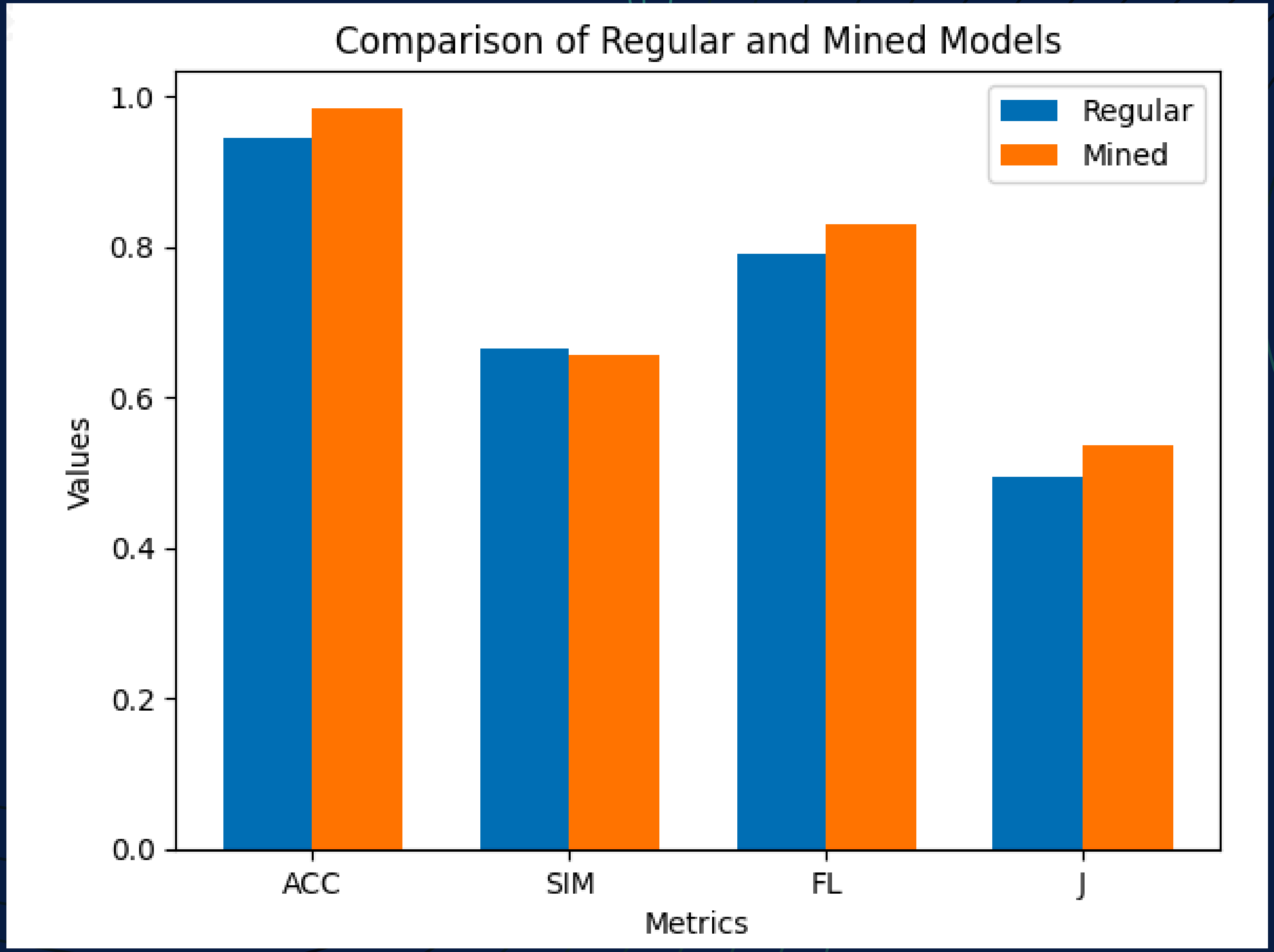
Regular



```
Calculating style of predictions
Some weights of the model checkpoint at SkolkovoInstitute/rob
- This IS expected if you are initializing RobertaForSequence
- This IS NOT expected if you are initializing RobertaForSequ
100%|██████████| 313/313 [18:07<00:00,  3.48s/it]
Calculating BLEU similarity
Calculating similarity by Wieting subword-embedding SIM model
100%|██████████| 313/313 [00:03<00:00, 90.28it/s]
| Model | ACC | SIM |

| ----- | --- | --- |

Mined|0.9840|0.6557|

| Model | ACC | SIM | FL | J | BLEU |

| ----- | --- | --- | -- | - | ---- |

Mined|0.9840|0.6557|0.8300|5355.0384|0.4528|
```

Mined

Results


Comparison of Regular and Mined Models
Regular
Mined
Values
1.0
0.8
0.6
0.4
0.2
0.0
ACC
SIM
FL
J
Metrics

Results

Accuracy (ACC):
The Mined model shows higher accuracy (0.9840) compared to the Regular model (0.9432). This suggests that the Mined model is more accurate in its task.

Similarity (SIM):
The SIM values for both models are relatively close, with the Regular model having a slightly higher SIM (0.6632) compared to the Mined model (0.6557). This implies that the models are quite similar in this metric.

Feature Length (FL):
The Mined model has a higher FL (0.8300) than the Regular model (0.7900). This indicates that the Mined model might use more features or have a more complex representation.

- The **Mined model has higher accuracy** and uses more features, which might contribute to its improved performance. This can also indicate that the Mined model could be more complex in terms of resource usage and computational requirements.

Qualitative Demonstration

To demonstrate the capabilities of our Gedi model, we developed an adapter that seamlessly integrates our trained paraphraser model with our generative gpt2-medium-based discriminative model. This adapter leverages the **generate()** function from the transformers.**Mixin** module within the Hugging Face Transformers library.

Thank You

Github Link:
https://github.com/nandithamerugu/Honours_Final_Project.git