Neural Network for Creating Diminutives

Project Results Presentation

About the Project

The goal of the project was to train the model so that when given a noun, it would return its diminutive.

Due to the lack of a ready dataset and the limited number of diminutives in the SJP, we tried using ChatGPT, but most of the diminutives it provided were incorrect and even comical.

Ultimately, we decided to manually prepare a dataset based on a list of the most popular nouns from the SGJP website and our knowledge of creating diminutives.

In this way, we created a CSV file with over 500 noun-diminutive pairs.

The input data is at the end of the code.

Training Model

Due to the specifications of the task, we decided to use a generative network and chose the T5 model.

Since we prepared the data ourselves in CSV format, it was enough to load it as a dataframe and ensure the appropriate columns:

- prefix - an empty column added, required by the model
- input_text - a column with base forms
- target_text - a column with diminutives

We split the dataset into:

- train_df (70%)
- eval_df (15%)
- test_df (15%)

Architecture

We chose the specific architecture of the T5 model based on the trial and error method. All the models we used are available on the Hugging Face website. Additionally, they had to be compatible with T5. We used Simple Transformers.

The models we used will be listed later in the presentation.

Model Training

Polish Language Models

Initially, we used models trained on Polish corpora:
amu-cai/polemma-base oraz *Voicelab/vlt5-base-keywords*

However, they were performing very poorly. The train_loss, at best, dropped to 3.18, but the model was still overfitted.

Moreover, the F1 scores and accuracy for the test data were 0.

Accuracy is: 0.0
F1 measure is: 0.0

Polish Language Models

- marcus2000/polish_reansliterator_T5

The only model among the "Polish" models that achieved non-zero F1 scores and accuracy was *marcus2000/polish_reansliterator_T5.*

```
Accuracy is: 0.012195121951219513
F1 measure is: 0.006211180124223602
```

Although it is a model trained on Polish corpora, adding the special_tokens parameter with Polish characters significantly improved its results.

```
model_type = "t5"
model_name = "marcus2000/polish_transliterator_T5"
special_tokens = ['ą', 'ę', 'ć', 'ł', 'ń', 'ó', 'ś', 'ż', 'ź']
```

```
    'special_tokens_list': special_tokens,
```

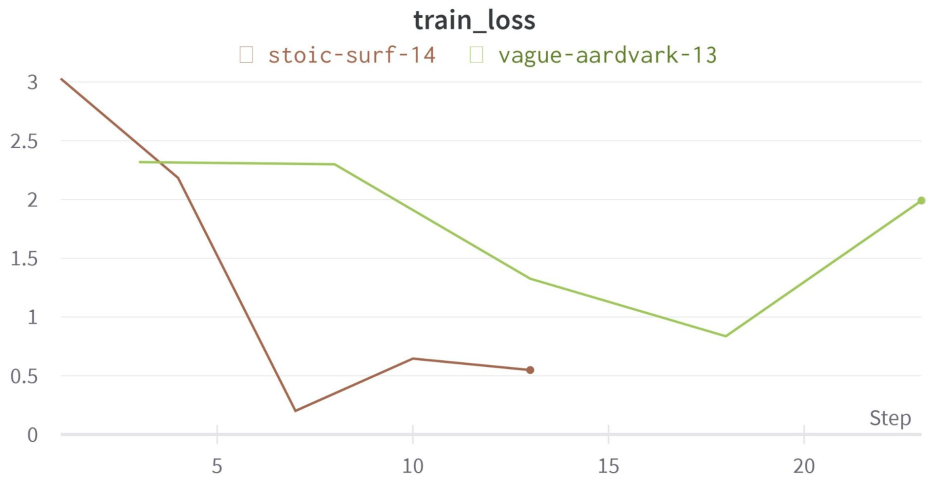
```
Accuracy is: 0.2926829268292683
F1 measure is: 0.17142857142857143
```

Polish Language Models

- marcus2000/polish_reansliterator_T5

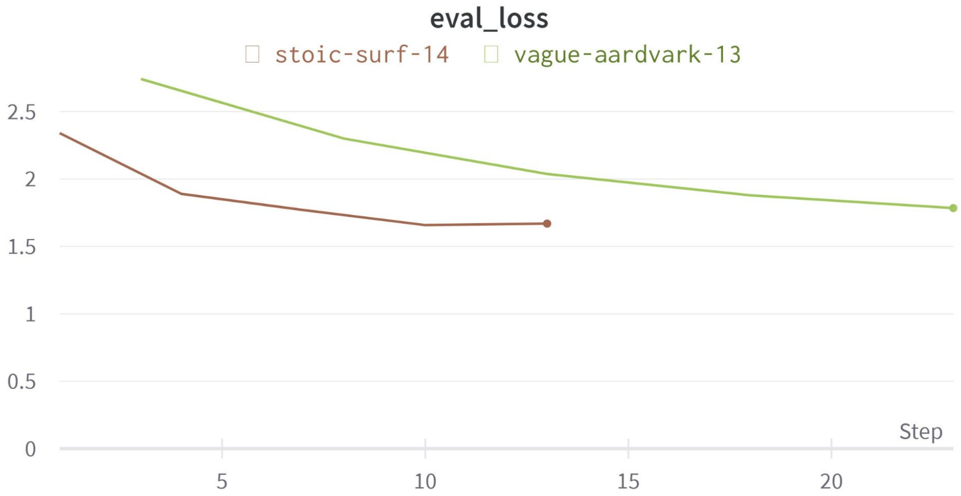
Unfortunately, we noticed that this model very often returned the letter “ł” in place of special characters, and in addition, added a space between this character.

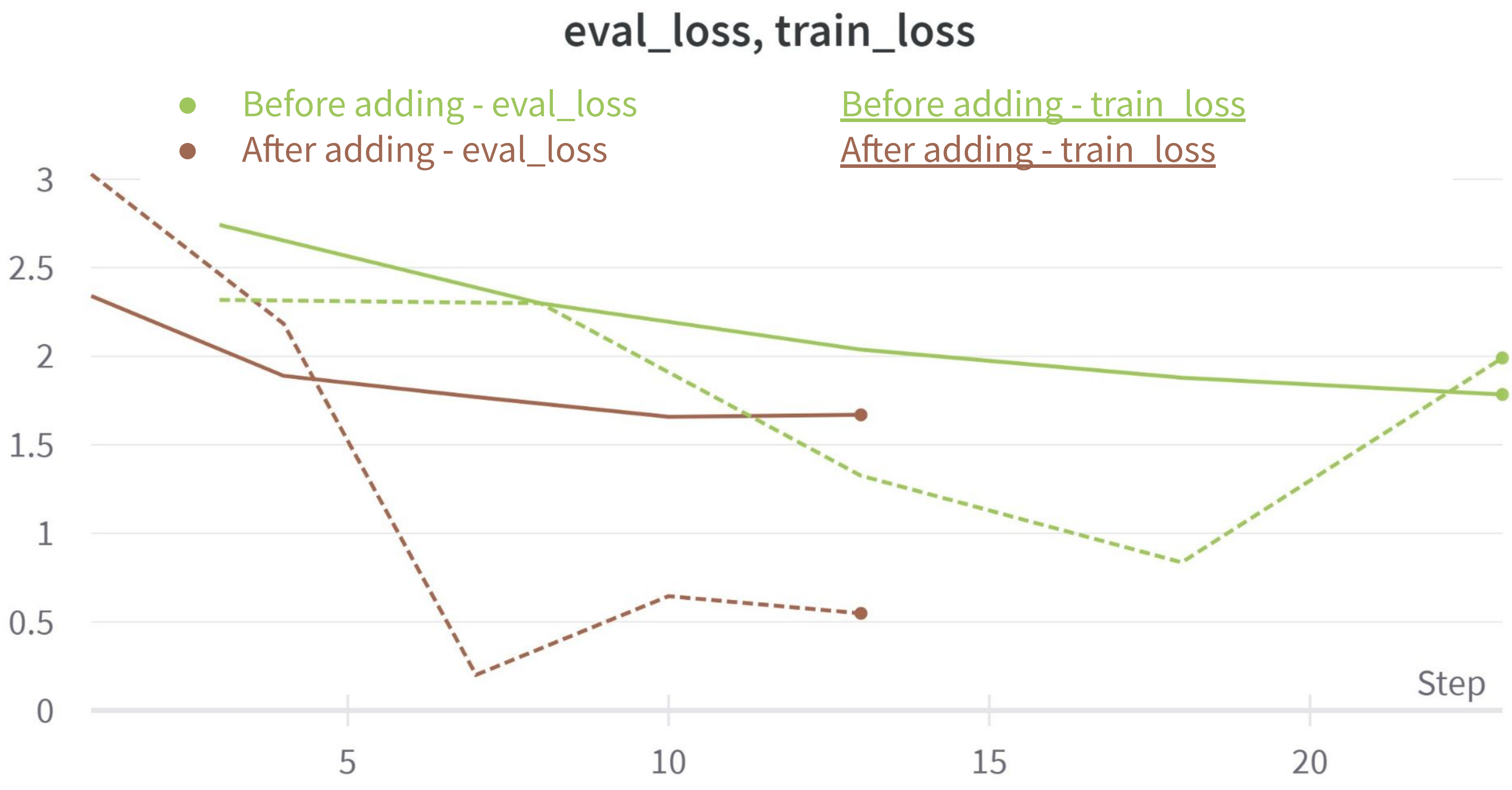
```
Diminutive of hasło: has ł ko
Diminutive of wystawa: wystawa
Diminutive of region: Region
Diminutive of uśmiech: u ł mieczek
Diminutive of godzina: godzinka
Diminutive of pieniądz: pieni ł ł dzka
Diminutive of nos: nok
Diminutive of remont: remontek
Diminutive of żart: ł artek
Diminutive of smak: smakzek
Diminutive of próba: próbka
Diminutive of kostka: kostka
```



Before adding special_tokens

After adding special_tokens




eval_loss, train_loss
Before adding - eval_loss
After adding - eval_loss
Before adding - train_loss
After adding - train_loss
3
2.5
2
1.5
1
0.5
0
5
10
15
20
Step

google/flan-t5-large / …-xl / …-xxl

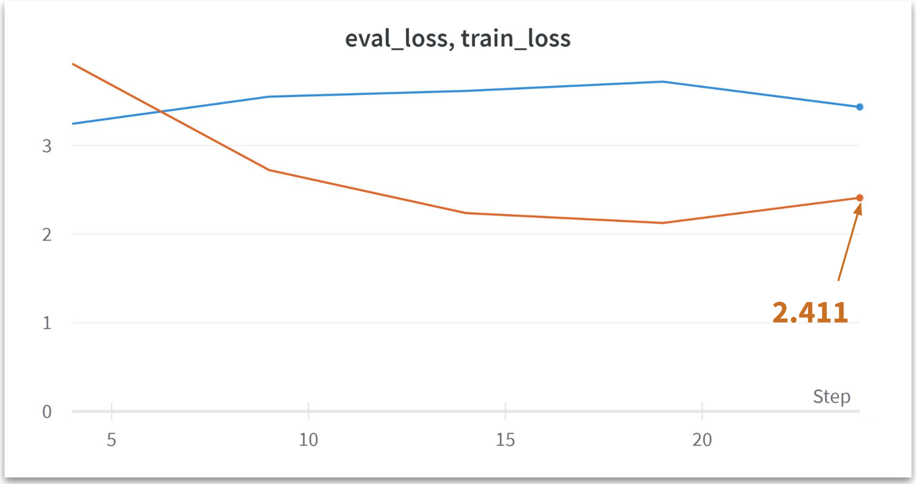
These models used too much system RAM, causing Google Colab to crash and terminate the session before completion. Therefore, we couldn't use such powerful models.

google/flan-t5-small ...mt5-small

The advantage of these models was relatively faster processing. However, their results were quite poor - the train_loss at best reached 2.41. Perhaps recalculating the models for a larger number of epochs would improve the results.

mt5-small

flan-t5-small

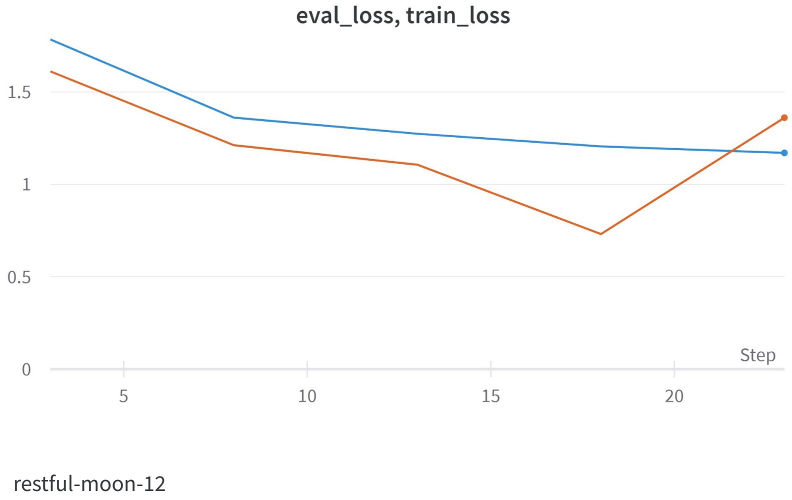


google/flan-t5-base

We worked the most with this model. It provided the most promising results, but the execution time for 5 epochs was over 2 hours. The following slides show the steps to improve the model.

google/flan-t5-base

Approach 1




```python
model_type = "t5"
model_name = "google/flan-t5-base"

train_args = {
    'evaluate_during_training': True,
    'num_train_epochs': 5,
    'save_eval_checkpoints': False,
    'train_batch_size': 2,
    'eval_batch_size': 2,
    'overwrite_output_dir': True,
    "reprocess_input_data": True,
    "max_seq_length": 128,
    "save_steps": -1,
    "use_multiprocessing": False,
    "fp16": False,
    'wandb_project': "Deminutywy",
    'learning_rate': 1e-5,
}

model = T5Model('t5', model_name, args=train_args, use_cuda=False)


model.train_model(train_df, eval_data=eval_df)
```

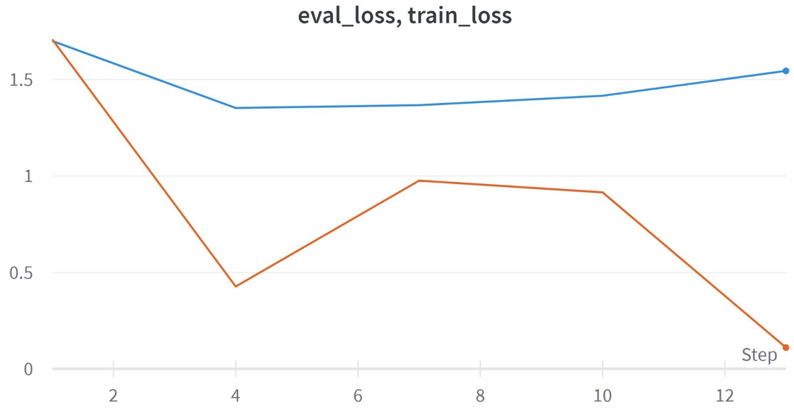
We noticed a problem with Polish character detection - no prediction contained any Polish character

```python
[26] f1 = f1_score(y_true=test_df["target_text"], y_pred=predictions, average="macro")
     accuracy = accuracy_score(y_true=test_df["target_text"], y_pred=predictions)

     print('Accuracy is:', accuracy)
     print('F1 measure is:', f1)
```

```
Accuracy is: 0.18292682926829268
F1 measure is: 0.10067114093959731
```

google/flan-t5-base

Approach 2



splendid-field-15

Here we tried adding special tokens to the model so that it would take Polish characters into account. However, a problem arose that even though the predicates contained Polish characters, each Polish character appeared as “ ł ” (including spaces)

```
model_type = "t5"
model_name = "google/flan-t5-base"
special_tokens = ['ą', 'ę', 'ć', 'ł', 'ń', 'ó', 'ś', 'ż', 'ź']

train_args = {
    'evaluate_during_training': True,
    'num_train_epochs': 5,
    'save_eval_checkpoints': False,
    'train_batch_size': 4,
    'eval_batch_size': 4,
    'overwrite_output_dir': True,
    "reprocess_input_data": True,
    "max_seq_length": 128,
    "save_steps": -1,
    "use_multiprocessing": False,
    "fp16": False,
    'wandb_project': "Deminutywy",
    'learning_rate': 3e-4,
    'special_tokens_list': special_tokens,
}

model = T5Model('t5', model_name, args=train_args, use_cuda=False)

model.train_model(train_df, eval_data=eval_df)
```

Raw Score:

```
Accuracy is: 0.32926829268292684
F1 measure is: 0.19708029197080293
```

The result after removing spaces:

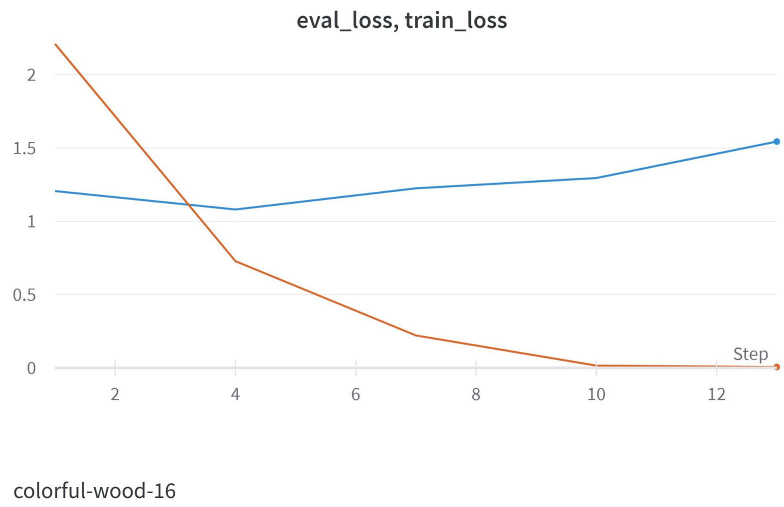
```
Accuracy is: 0.3780487804878049
F1 measure is: 0.23308270676691728
```

The result without Polish characters:

```
Accuracy is: 0.4268292682926829
F1 measure is: 0.2713178294573643
```

google/flan-t5-base

Approach 3



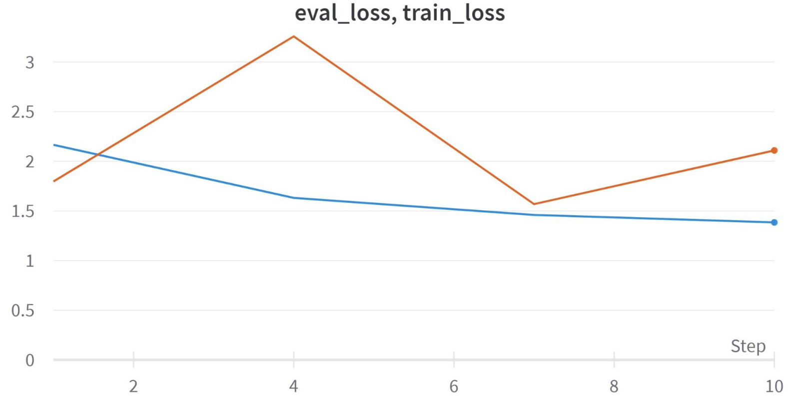

```
model_type = "t5"
model_name = "google/flan-t5-base"

train_args = {
    'evaluate_during_training': True,
    'num_train_epochs': 5,
    'save_eval_checkpoints': False,
    'train_batch_size': 4,
    'eval_batch_size': 4,
    'overwrite_output_dir': True,
    "reprocess_input_data": True,
    "max_seq_length": 128,
    "save_steps": -1,
    "use_multiprocessing": False,
    "fp16": False,
    'wandb_project': "Deminutywy",
    'learning_rate': 3e-4,
}

model = T5Model('t5', model_name, args=train_args, use_cuda=False)

model.train_model(train_df, eval_data=eval_df)
```

In this approach we removed all Polish characters from the dataset and train the model on such words.
The train_loss decreased quickly, but the eval_loss slowly increased from the beginning.

```
Accuracy is: 0.4024390243902439
F1 measure is: 0.25190839694656486
```

google/flan-t5-base

Approach 4

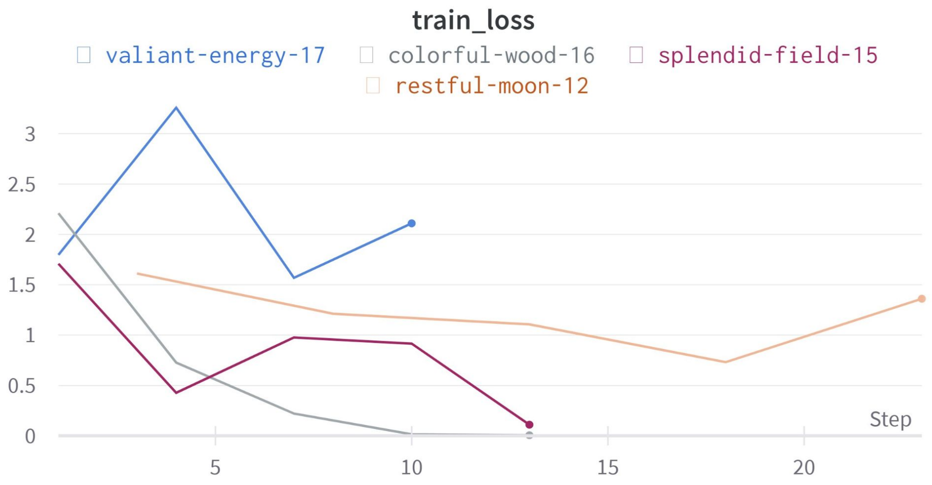


valiant-energy-17

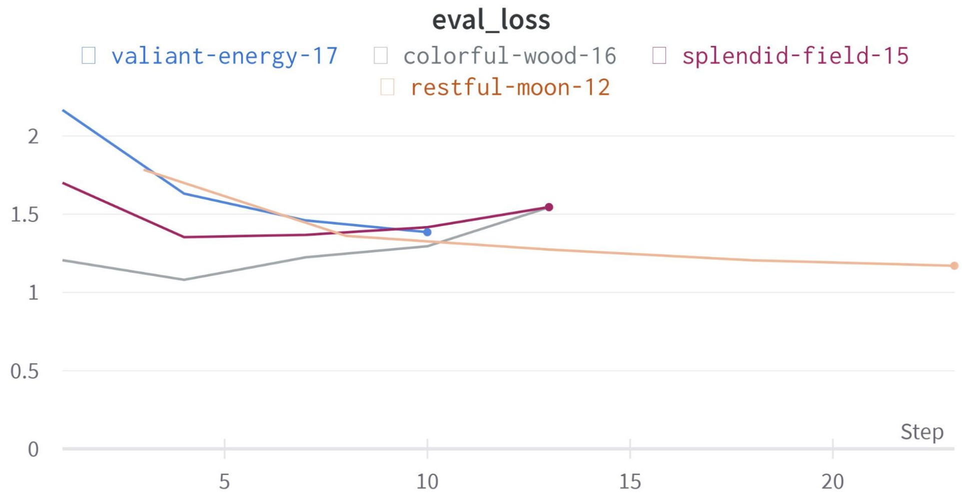
```
model_type = "t5"
model_name = "google/flan-t5-base"

train_args = {
    'evaluate_during_training': True,
    'num_train_epochs': 4,
    'save_eval_checkpoints': False,
    'train_batch_size': 4,
    'eval_batch_size': 4,
    'overwrite_output_dir': True,
    "reprocess_input_data": True,
    "max_seq_length": 128,
    "save_steps": -1,
    "use_multiprocessing": False,
    "fp16": False,
    'wandb_project': "Deminutywy",
    'learning_rate': 1e-5,
}

model = T5Model('t5', model_name, args=train_args, use_cuda=False)

model.train_model(train_df, eval_data=eval_df)
```

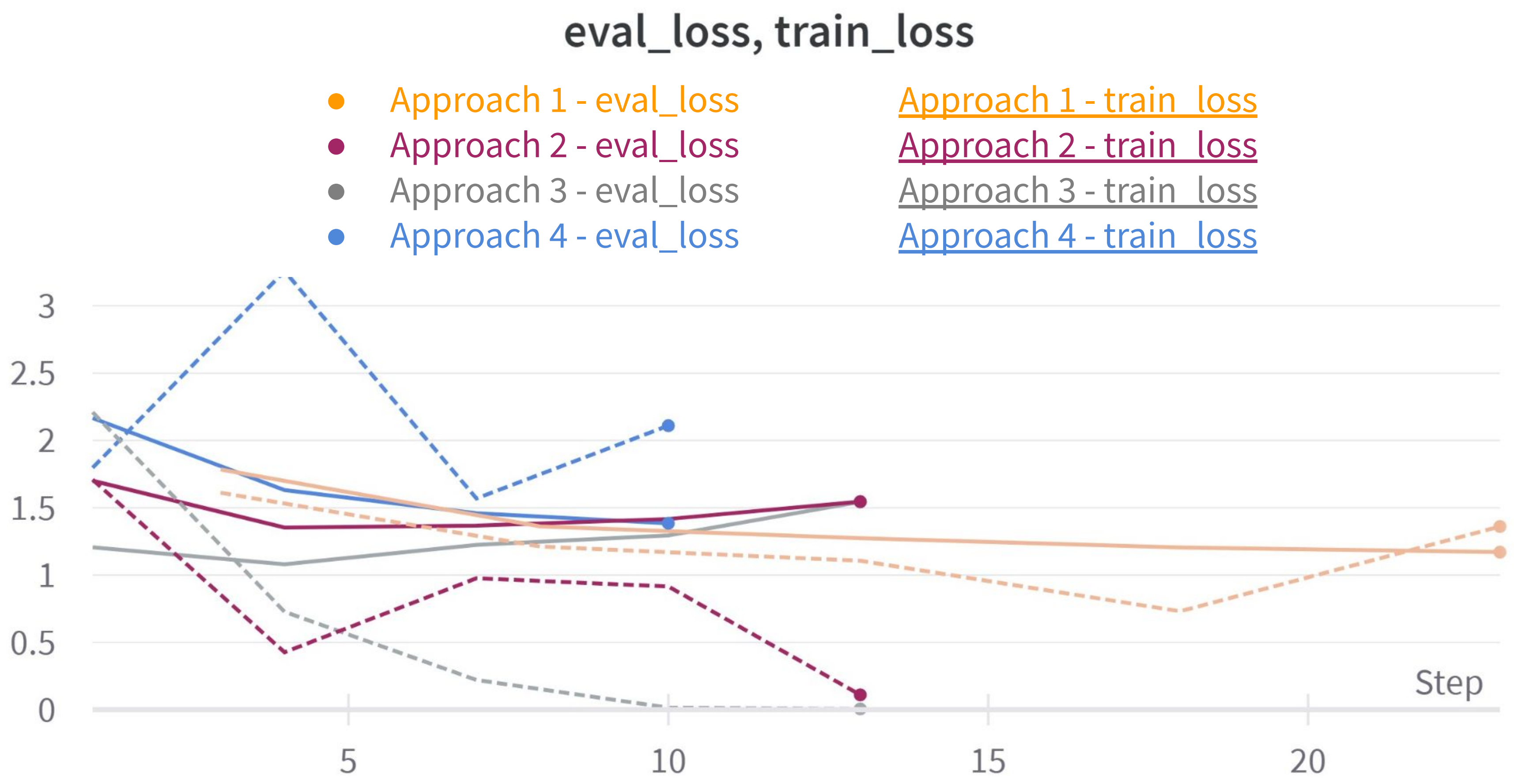
This approach differs from the previous one only in the reduced value of learning_rate.
We also lowered the number of epochs by 1 to prevent possible overtraining.
However, we received one of the worst results.

```
Accuracy is: 0.10975609756097561
F1 measure is: 0.05806451612903226
```



- Approach 1
- Approach 2
- Approach 3
- Approach 4





eval_loss, train_loss
Approach 1 - eval_loss
Approach 2 - eval_loss
Approach 3 - eval_loss
Approach 4 - eval_loss
Approach 1 - train_loss
Approach 2 - train_loss
Approach 3 - train_loss
Approach 4 - train_loss
3
2.5
2
1.5
1
0.5
0
Step
5
10
15
20

Conclusions

One of the main problems during model training was Polish characters. The Hugging Face models, which even stated in their descriptions that they included the Polish language, did not handle them well during training.

The second problem was the limited RAM in Google Colab. For this reason, we had to reduce the train_batch_size and eval_batch_size to a maximum of 4. We also limited the number of epochs to 5 because training the model sometimes took 5 hours. This prevented us from trying many times with changed parameters.

Conclusions

In the best case, we achieved an accuracy of over 40% and an F1 score of over 25%. Unfortunately, this was achieved for a model trained and then tested on data without Polish characters. For a model that took Polish characters into account, we achieved an accuracy of about 33% and an F1 score of about 20%.

A relatively large proportion of the generated diminutives that did not contain any Polish characters were correct.

In the future, this result could be improved by recalculating the model for a larger number of epochs and carefully selecting the learning_rate.

Thank you for you attention

Julia Akahori
s20936

Ignacy Bok
s20883