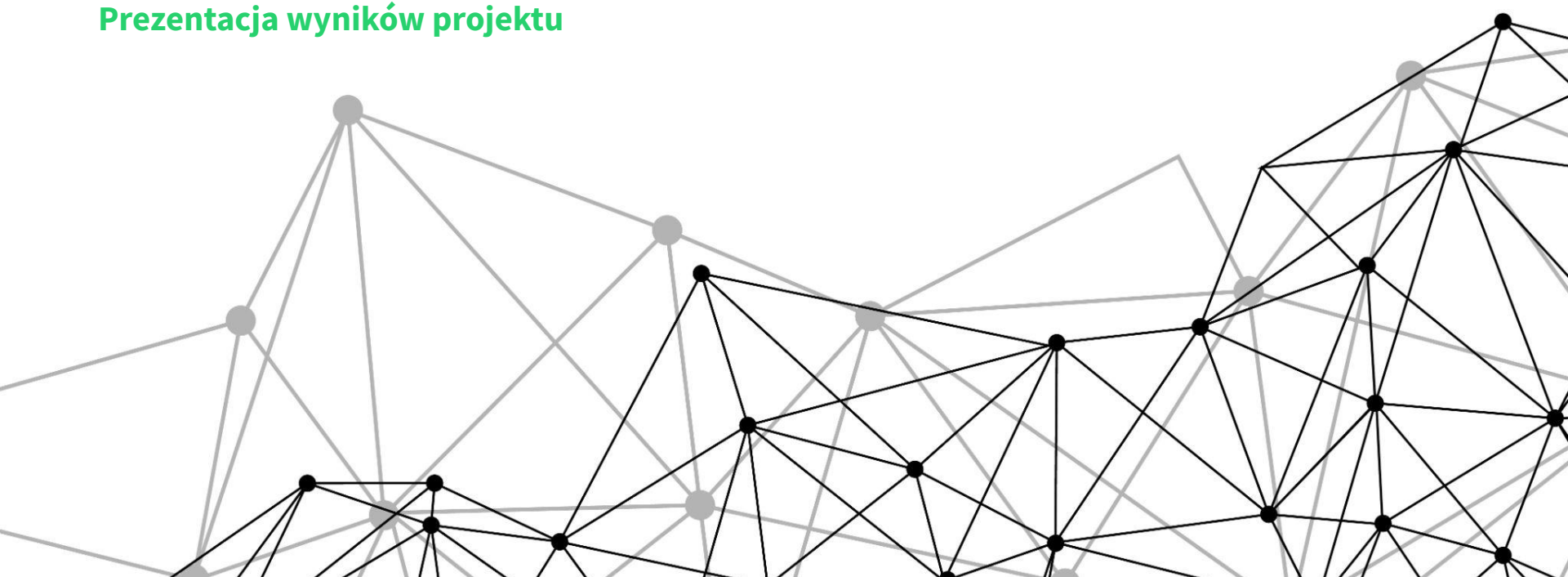
Sieć neuronowa tworząca deminutywy

Prezentacja wyników projektu



O projekcie

Celem projektu było wytrenowanie modelu tak, aby po podaniu rzeczownika, zwrócił jego zdrobnienie.

Z powodu braku gotowego zestawu danych oraz ograniczonej liczby zdrobnień w SJP, próbowaliśmy skorzystać z Chat'u GPT, ale większość deminutywów, które podał, była niepoprawna i wręcz komiczna.

Ostatecznie postanowiliśmy przygotować własnoręcznie zestaw danych na podstawie listy najpopularniejszych rzeczowników ze strony SGJP oraz własnej wiedzy na temat tworzenia deminutywów.

W ten sposób przygotowaliśmy plik csv z ponad 500 pozycjami rzeczownik-zdrobnienie.

Dane wejściowe znajdują się na końcu kodu.

Model treningowy

Ze względu na specyfikację zadania, zdecydowaliśmy się skorzystać z sieci generatywnej i wybraliśmy model T5.

Ponieważ dane przygotowaliśmy sami w formacie CSV, wystarczyło je wczytać jako dataframe i zadbać o odpowiednie kolumny:

- prefix - dodana pusta kolumna, wymagana przez model
- input_text - kolumna z formami podstawowymi
- target_text - kolumna ze zdrobnieniami

Dataset podzieliliśmy na:

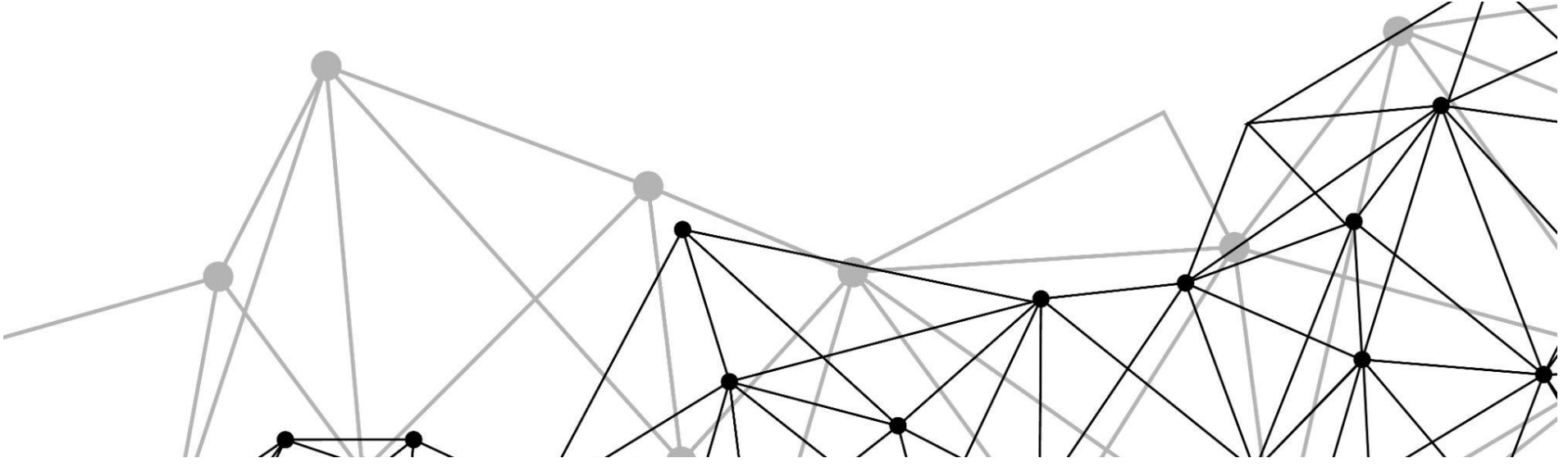
- train_df (70%)
- eval_df (15%)
- test_df (15%)

Architektura

Konkretną architekturę modelu T5 wybraliśmy na podstawie metody prób i błędów. Wszystkie wykorzystane przez nas modele są dostępne na stronie Hugging Face. Dodatkowo musiały być kompatybilne z T5. Korzystaliśmy z Simple Transformers.

Wykorzystane modele będą wymienione w dalszej części prezentacji.

Trening modeli



Modele z językiem polskim

Na początku korzystaliśmy z modeli wytrenowanych na podstawie polskich korpusów: *amu-cai/polemma-base* oraz *Voicelab/vlt5-base-keywords*

Jednak, osiągały one bardzo słabe wyniki. Train_loss w najlepszym przypadku spadała do 3.18, natomiast i tak dochodziło do przetrenowania modelu.

Co więcej, miary F1 i dokładność dla danych testowych wynosiły 0

```
Accuracy is: 0.0
F1 measure is: 0.0
```

Modele z językiem polskim

- marcus2000/polish_reansliterator_T5

Niezerowe wyniki dla miar F1 i dokładności spośród “polskich” modeli zwrócił dopiero model *marcus2000/polish_reansliterator_T5.*

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

Mimo, iż jest to model wytrenowany w oparciu o polskie korpusy, dodanie parametru special_tokens z polskimi znakami znacząco poprawiło jego wyniki

```
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
```

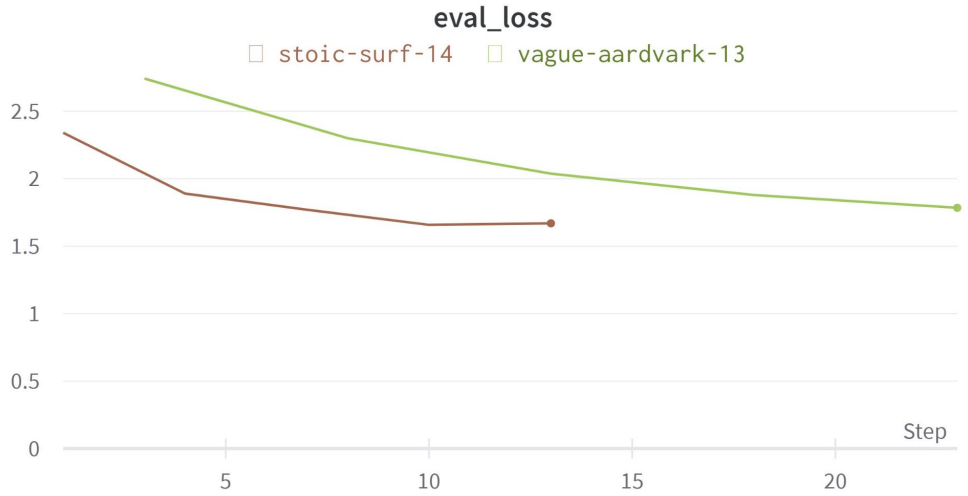
Modele z językiem polskim

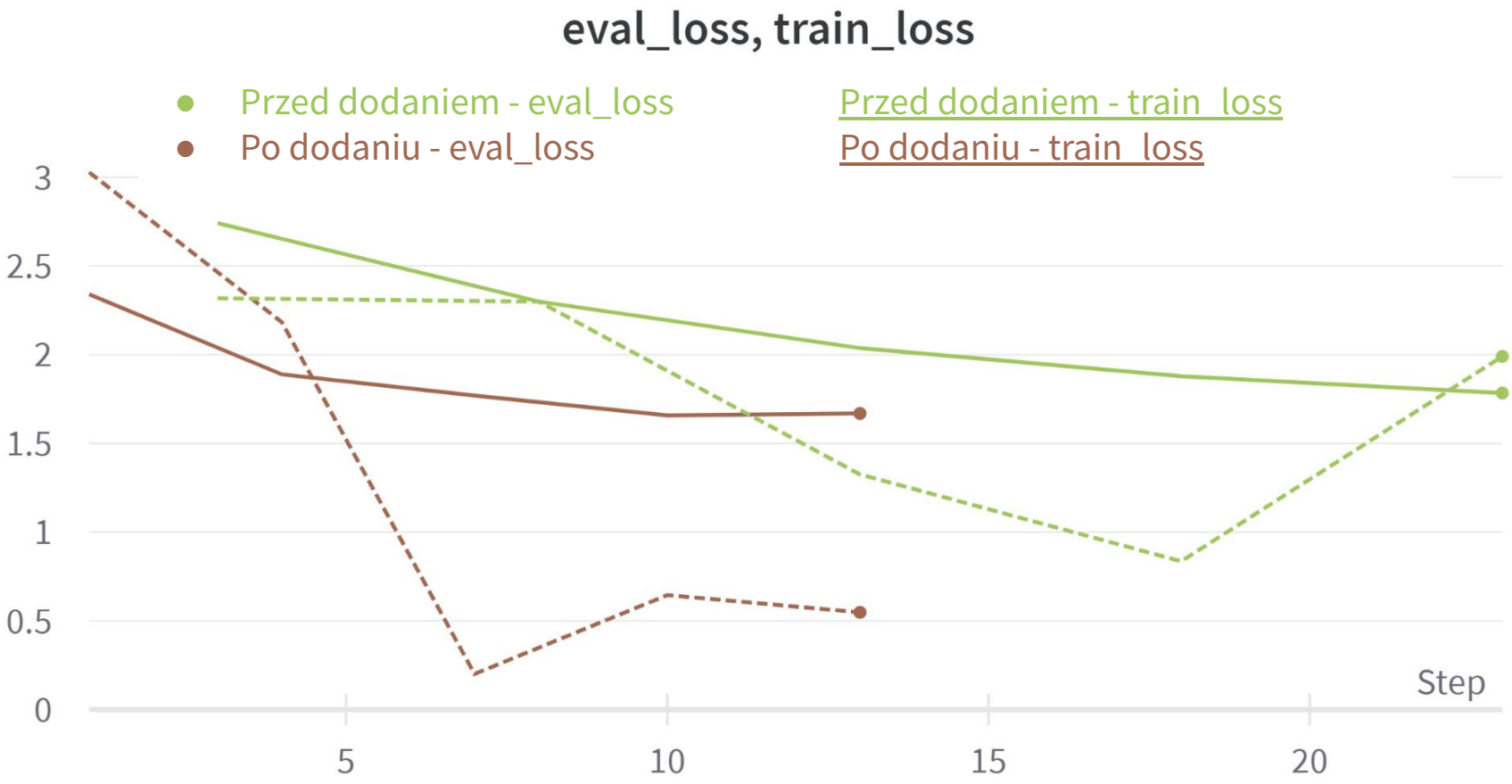
- marcus2000/polish_reansliterator_T5

Niestety, zauważyliśmy, że ten model bardzo często w miejscu znaków specjalnych zwracał literę “ł”, a w dodatku dodawał spację pomiędzy tym znakiem

```
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
```

- Przed dodaniem special_tokens
- Po dodaniu special_tokens




eval_loss, train_loss
Przed dodaniem - eval_loss
Po dodaniu - eval_loss
Przed dodaniem - train_loss
Po dodaniu - train_loss
3
2.5
2
1.5
1
0.5
0
5
10
15
20
Step

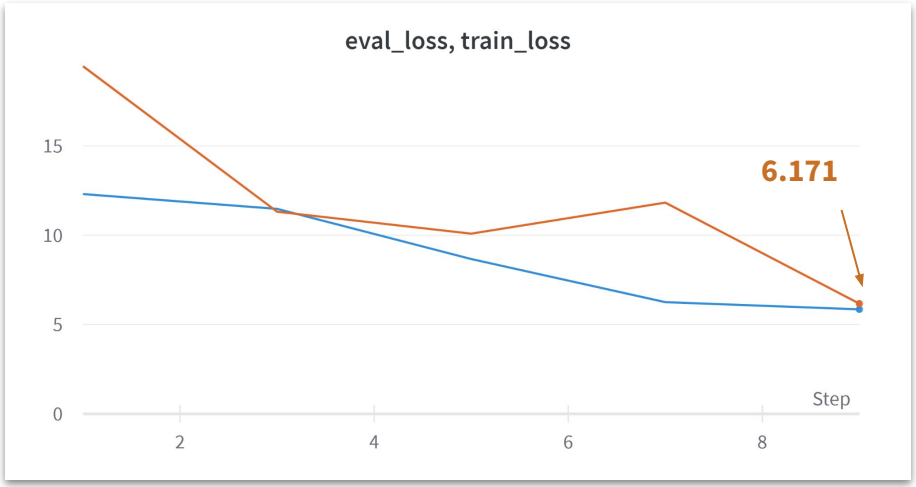
google/flan-t5-large / ...-xl / ...-xxl

Te modele zużywały zbyt wiele systemowej pamięci RAM, przez co Google Colab przed końcem przerywał sesję i kończył działanie z informacją o awarii. Nie mogliśmy więc korzystać z tak potężnych modeli.

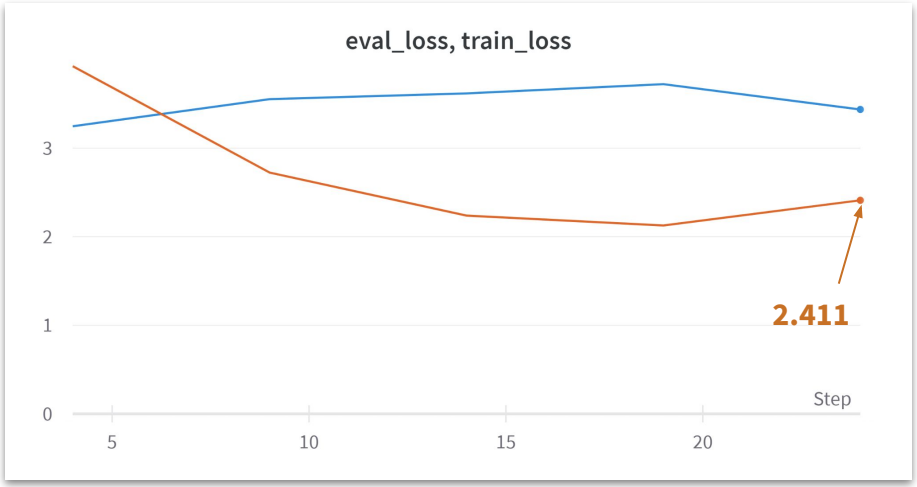
google/flan-t5-small ...mt5-small

Zaletą tych modeli było stosunkowo szybsze przeliczanie się. Jednak ich wyniki były dość słabe - train_loss w najlepszym przypadku osiągnęła 2.41. Być może, przeliczenie modeli dla większej ilości epok poprawiłoby rezultaty.

mt5-small



flan-t5-small

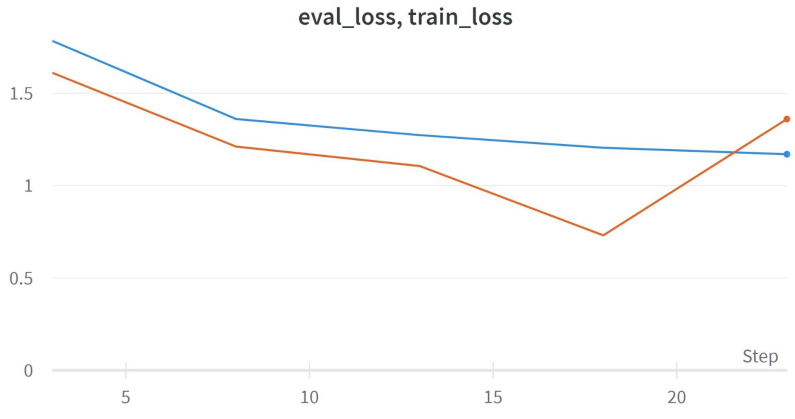


google/flan-t5-base

Z tym modelem najwięcej pracowaliśmy. Dawał on najbardziej obiecujące wyniki, jednak czas wykonywania 5 epoch wynosił ponad 2 godziny. W kolejnych slajdach pokażemy kolejne kroki udoskonalania modelu.

google/flan-t5-base

Podejście 1



restful-moon-12

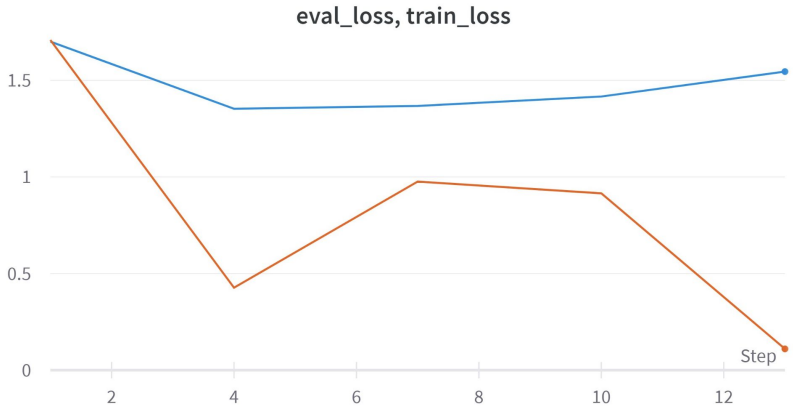
```
model_type = "t5"
model_name = "google/flan-t5-base"

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)
```

Zauważyliśmy problem z wykrywaniem znaków polskich - żadna predykcja nie zawierała jakiegokolwiek znaku polskiego

```
[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

Podejście 2



splendid-field-15

```
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)
```

Tu spróbowaliśmy dodać do modelu specjalne tokeny, żeby uwzględniał on polskie znaki. Powstał jednak problem, że pomimo iż predykcje zawierały znaki polskie, każdy znak polski występował jako “ ł ” (z uwzględnieniem spacji)

Surowy wynik:

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

Wynik po usunięciu spacji:

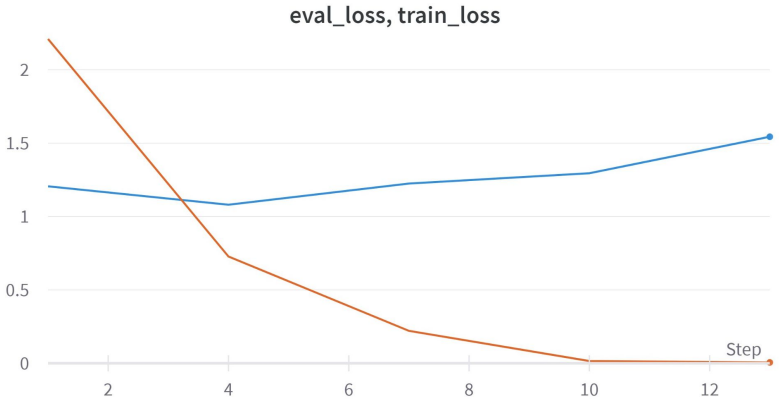
```
Accuracy is: 0.3780487804878049
F1 measure is: 0.23308270676691728
```

Wynik bez polskich znaków:

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

google/flan-t5-base

Podejście 3



colorful-wood-16

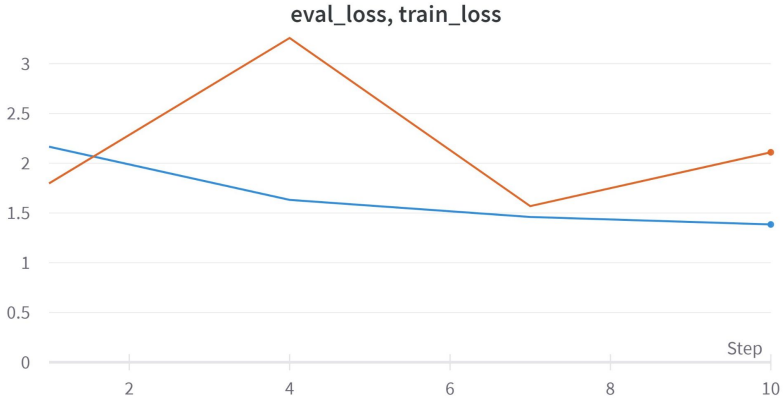
```
model_type = "t5"
model_name = "google/flan-t5-base"

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)
```

To podejście polegało na usunięciu wszystkich polskich znaków z datasetu i trenowaniu modelu na takich słowach.
Widać jak szybko train_loss malał, ale eval_los prawie od samego początku powoli rósł.

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

google/flan-t5-base

Podejście 4

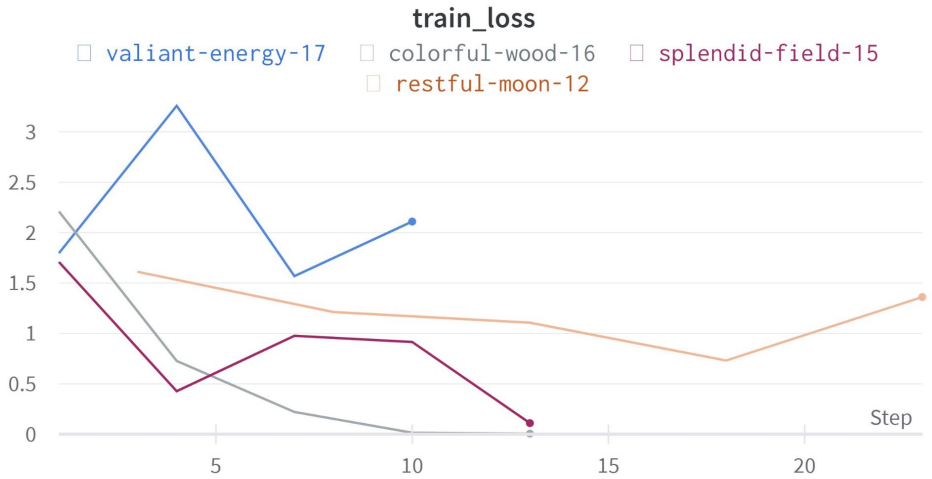


valiant-energy-17

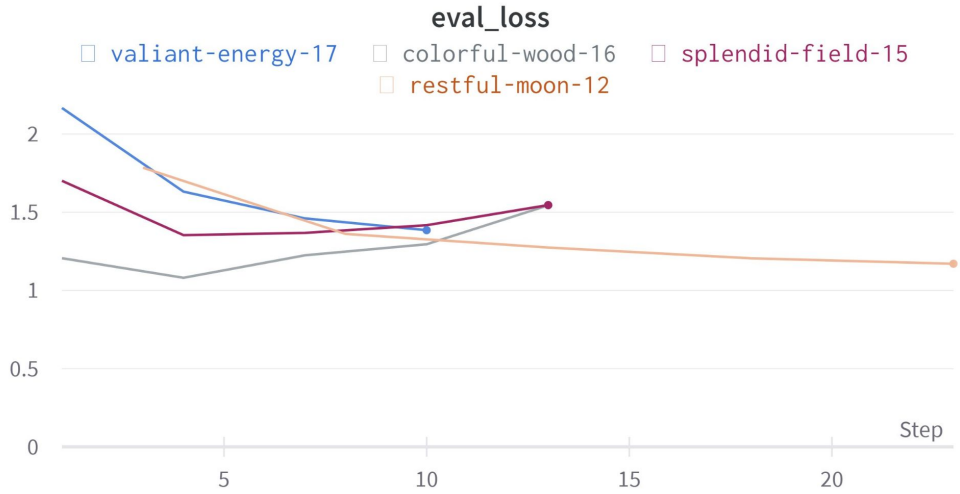
```
model_type = "t5"
model_name = "google/flan-t5-base"

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)
```

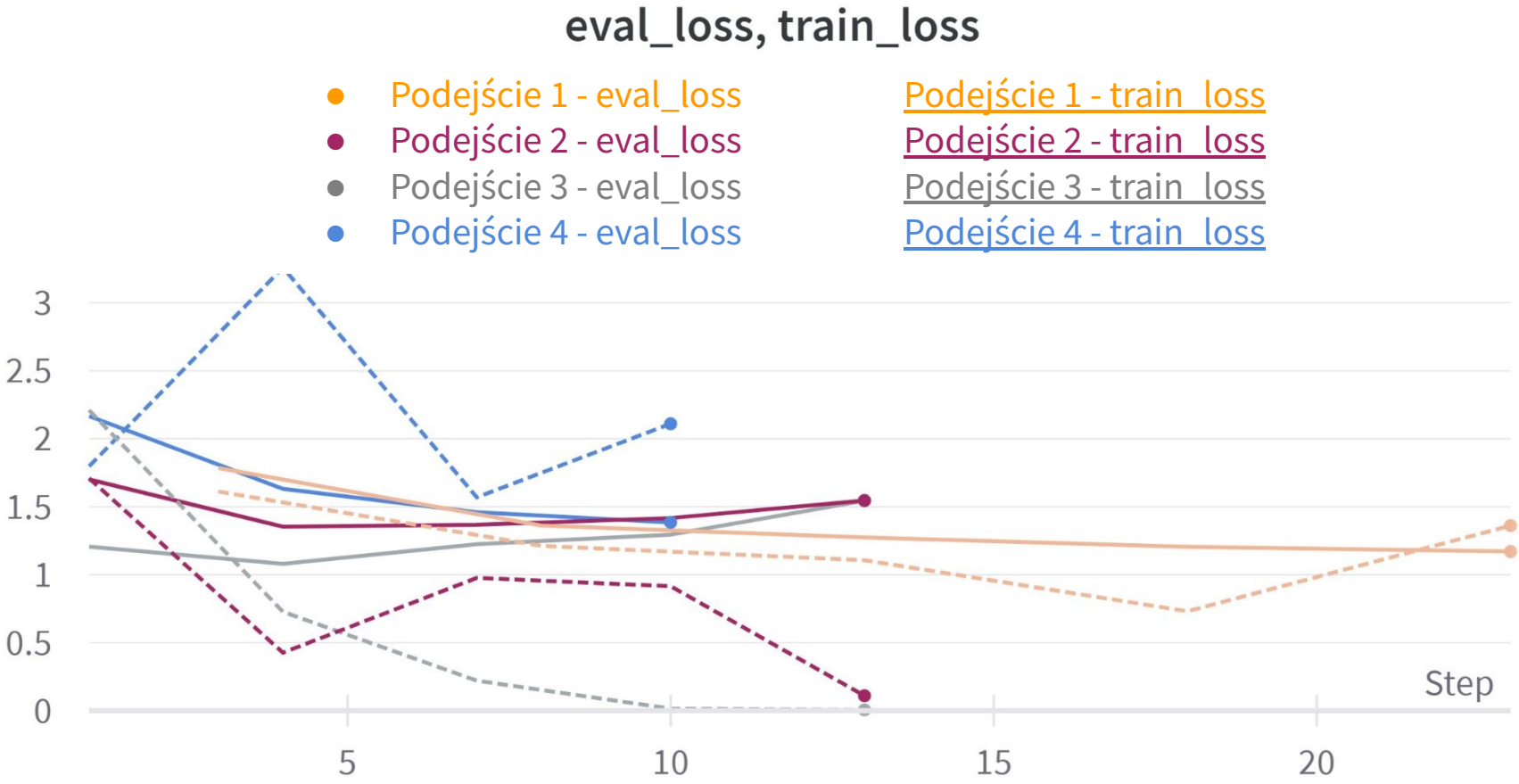
To podejście od poprzedniego różni się jedynie zmniejszoną wartością learning_rate.
Obniżyliśmy też liczbę epoch o 1, aby zapobiec ewentualnemu przetrenowaniu.
Otrzymaliśmy jednak jeden z najgorszych wyników.

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



- Podejście 1
- Podejście 2
- Podejście 3
- Podejście 4




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

Wnioski

Jednym z głównych problemów podczas trenowania modelu były znaki polskie. Modele z Hugging Face, które nawet w opisie miały zapis, że uwzględniają język polski, w trenowaniu nie radziły sobie z nimi dobrze.

Drugim problemem była ograniczona pamięć RAM w Google Colab. Z tego powodu zmuszeni byliśmy do zmniejszenia train_batch_size oraz eval_batch_size do maksymalnie 4. Zdecydowaliśmy się również ograniczyć liczbę epok do 5, ponieważ trenowanie modelu trwało niekiedy 5h. Przez to nie mogliśmy wiele razy próbować ze zmienionymi parametrami.

Wnioski

W najlepszym przypadku udało się osiągnąć dokładność ponad 40% oraz miarę F1 ponad 25%. Niestety, osiągnięto je dla modelu wytrenowanego, a następnie testowanego na danych nie zawierających polskich znaków. Dla modelu biorącego pod uwagę polskie znaki osiągnięto dokładność = ok. 33% oraz miarę F1 = ok. 20%.

Stosunkowo duża część wygenerowanych deminutywów, które nie zawierały żadnych polskich znaków była poprawna.

W przyszłości można by ten wynik spróbować poprawić przeliczając model za większą liczbę epok, oraz precyzyjnie dobierając learning_rate.

Dziękujemy za uwagę

Julia Akahori
s20936

Ignacy Bok
s20883