Inżynieria Lingwistyczna

Projekt: Wykrywanie mowy nienawiści i cyberprzemocy za pomocą BERT

Michalina Kwolik s23922
Gabriela Olszewska s28652

Temat

Tematem projektu było użycie modelu językowego BERT do wykrywania mowy nienawiści oraz cyberprzemocy w Internecie. Dane do trenowania modelu zostały wzięte ze strony http://2019.poleval.pl/index.php/tasks/task6 i zawierały tweety pobrane z publicznych dyskusji na platformie twitter.com.

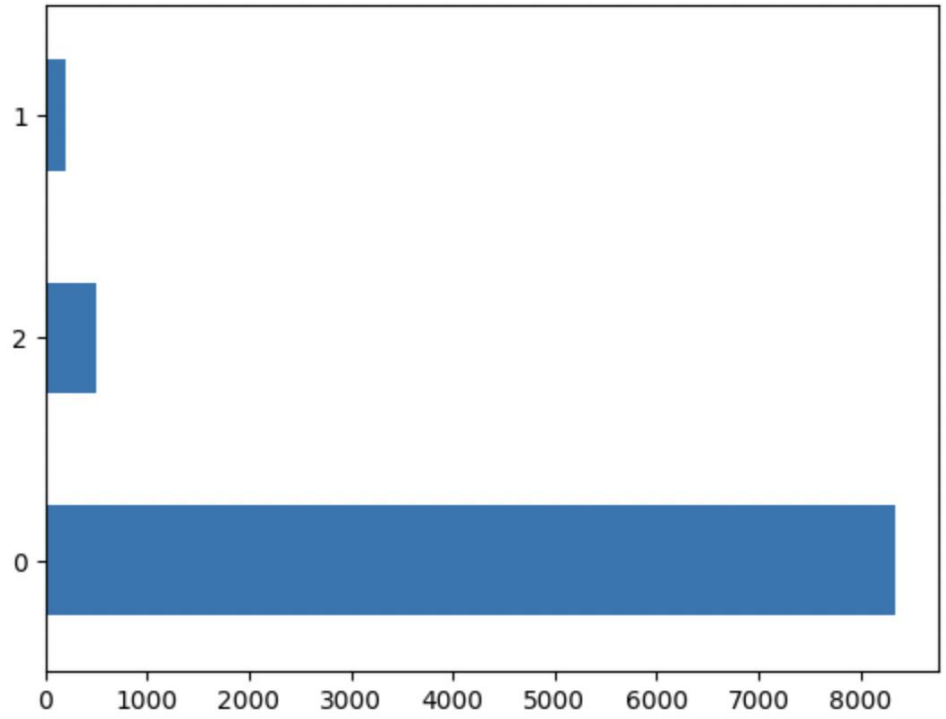
Wpisy były klasyfikowane w kategorii trzech klas, 0 - wydźwięk neutralny, 1 – cyberbullying oraz 2 – hatespeech. Został pobrany zestaw z danymi treningowymi (podzielony później na zestaw ewaluacyjny) oraz zestaw testowy.

Przygotowanie danych

1. Ściągnięcie dwóch zestawów danych (po dwa pliki każdy, jeden to tekst, drugi to etykiety)
2. Połączenie każdego zestawu w dataframe z kolumnami 'text' i 'labels'
3. Dataframe treningowy podzielony na zestaw **treningowy** liczący **9041** etykiet oraz na zestaw **ewaluacyjny** liczący **1000** etykiet
4. Dataframe z zestawem **testowym** pozostaje bez zmian (**1000** etykiet), przyjęty podział 80-10-10
5. Podjęta próba wyczyszczenia danych lecz przy późniejszym trenowaniu pojawia się błąd (próba obejmowała usunięcia słów "RT" i "@anonymized_account" oraz "emojis")

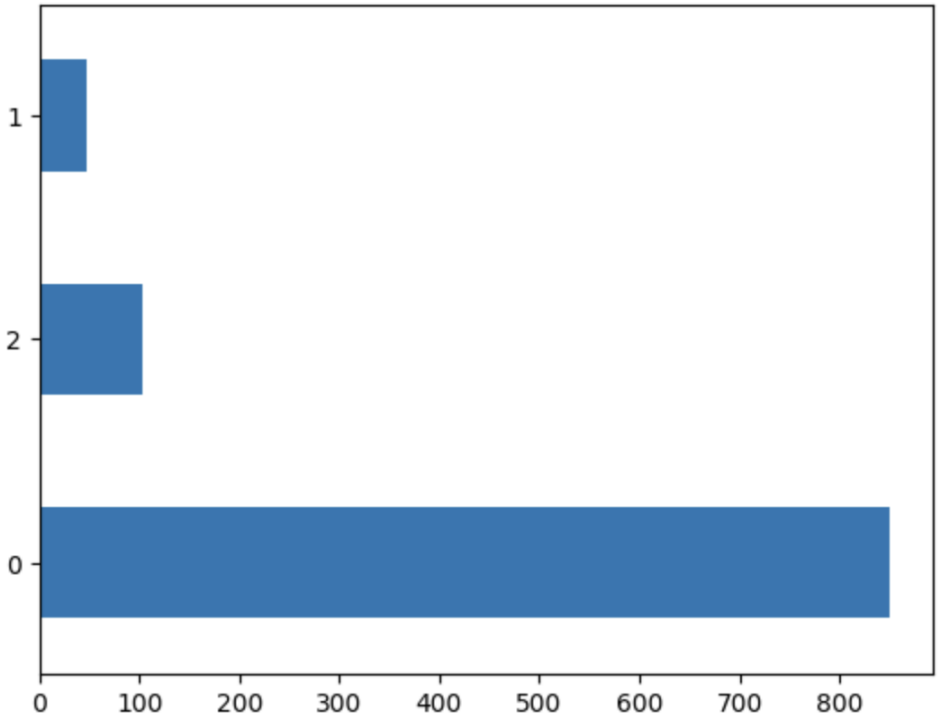
```
def preprocess_dataframe(df):
    df['text'] = df['text'].apply(lambda x: x.replace('@anonymized_account', ''))
    df['text'] = df['text'].apply(lambda x: x.replace('RT', ''))
    df['text'] = df['text'].apply(lambda x: re.sub(r'[^\w\s\d\s]+', '', x))

    return df

df = preprocess_dataframe(df)
```

Dane treningowe:



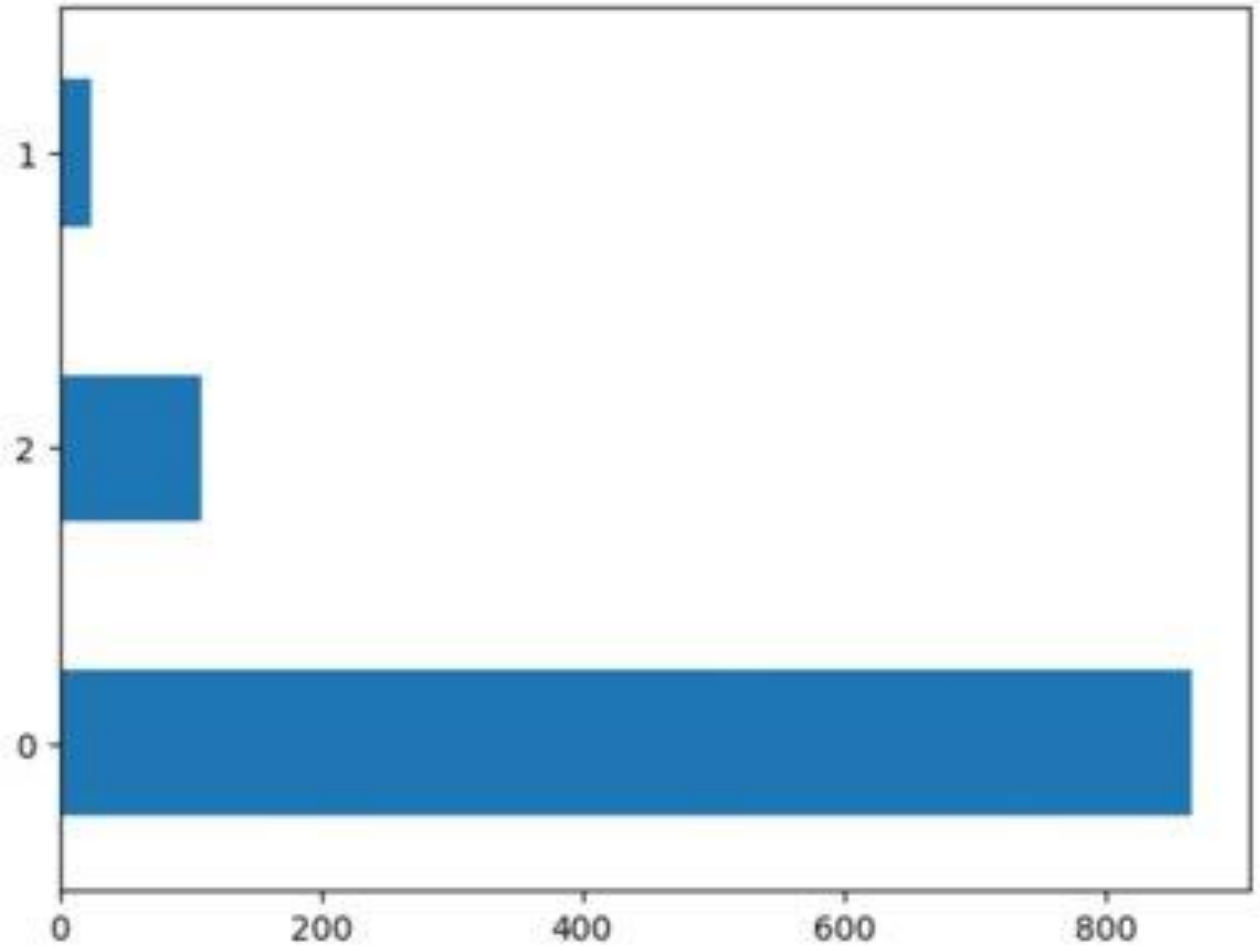

```
Value count for training data:
0    8340
2     495
1     206
Name: labels, dtype: int64
```

Dane ewaluacyjne:




```
Value count for evaluation data:
0    850
2    103
1     47
Name: labels, dtype: int64
```

Dane testowe:




```
Value count for test data:
0    866
2    109
1     25
Name: labels, dtype: int64
```

Wnioski

- Etykiety 1 i 2 w każdym dataset są w mniejszości (zwykle stanowią niecałe 10% wszystkich etykiet)
- Taki nierównomierny rozkład danych może wpływać na skuteczność modelu w wykrywaniu hatespeech lub cyberbullying
- Model może mieć tendencję do dominacji klasy 0 z powodu dużo większej liczby jej wystąpień
- Można ten problem rozwiązać za pomocą oversampling lub undersampling
- Nierównomierny rozkład klas może wpłynąć na jakość oceny modelu (np. wysoka precyzja dla klasy 0, a słabszy wynik dla klas 1 i 2)

Trenowanie modelu

- Do trenowania modelu wykorzystano model DistilBERT (mniejszy i szybszy niż BERT) z biblioteki Simple Transformers
- Wszystkie próby (udane i nieudane) zapisane były na Wandb

- Pierwsze cztery próby zakończyły się błędami gdy przechodziła prawie cała pierwsza epoka (po około godzinie). Wynikało to na początku z powodu kodu na czyszczenie danych (wspomniane wcześniej), a później problem pojawiał się z kodem na liczenie F1, accuracy, precision i recall. Mimo nieudanych prób, argumenty do trenowania były zmieniane żeby zobaczyć jak model zachowuje się w trakcie trenowania. Zmieniana była liczba epok (pierwsze cztery próby i tak nie dokończały pierwszej epoki), learning rate oraz training i evaluation batch size. Jedyne wykresy z tych przebiegów to training loss, global step oraz learning rate.

RUN 1
LR: 4e-5
EPOCHS: 1
TRAIN BATCH SIZE: 32
EVAL BATCH SIZE: 32

RUN 2
LR: 1e-5
EPOCHS: 1
TRAIN BATCH SIZE: 16
EVAL BATCH SIZE: 32

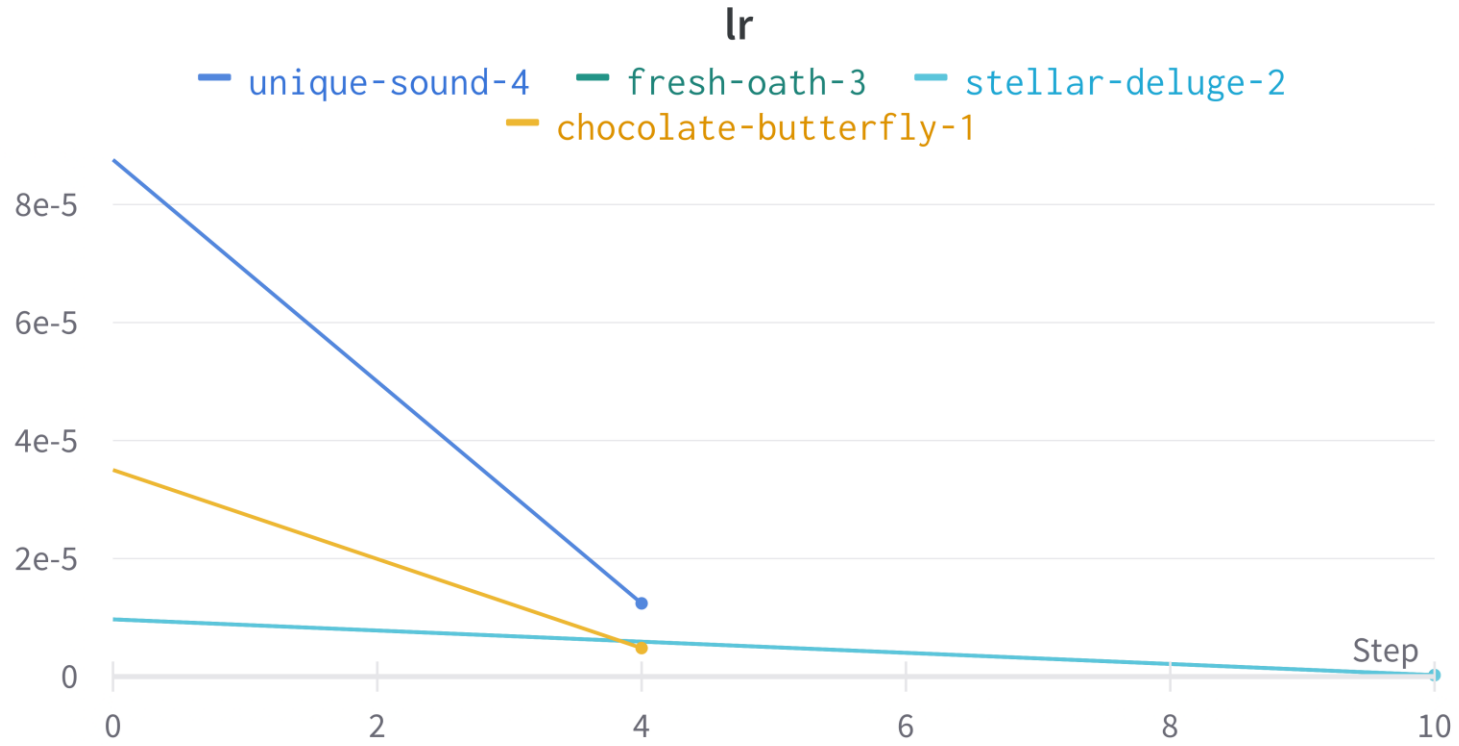
RUN 3
LR: 1e-5
EPOCHS: 1
TRAIN BATCH SIZE: 16
EVAL BATCH SIZE: 32

RUN 4
LR: 1e-5
EPOCHS: 2
TRAIN BATCH SIZE: 32
EVAL BATCH SIZE: 32

```python
def metrics(labels, preds):
    accuracy = accuracy_score(labels, preds)
    precision = precision_score(labels, preds, average='weighted')
    recall = recall_score(labels, preds, average='weighted')
    f1 = f1_score(labels, preds, average='weighted')

    return {
        'accuracy': accuracy,
        'precision': precision,
        'recall': recall,
        'f1_score': f1
    }

#argumenty do treningu
train_args = {
    'learning_rate': 1e-3,
    'evaluate_during_training': True, #musi byc true
    'num_train_epochs': 2,
    'save_eval_checkpoints': False, #lepiej na false
    'train_batch_size': 32,
    'eval_batch_size': 32,
    'overwrite_output_dir': True, #outputs w plikach obok
    'wandb_project': "hatespeech_training"
}

#inicjalizacja wandb
wandb.init(project=train_args['wandb_project'])

#stworzenie modelu klasyfikacji
model = ClassificationModel('distilbert', 'distilbert-base-cased', num_labels=3, use_cuda=False, cuda_device=0, args=train_args)

#trenowanie modelu
model.train_model(training_df, eval_df = eval_df, eval_metrics = metrics)
result, model_outputs, wrong_predictions = model.eval_model(test_df)
```

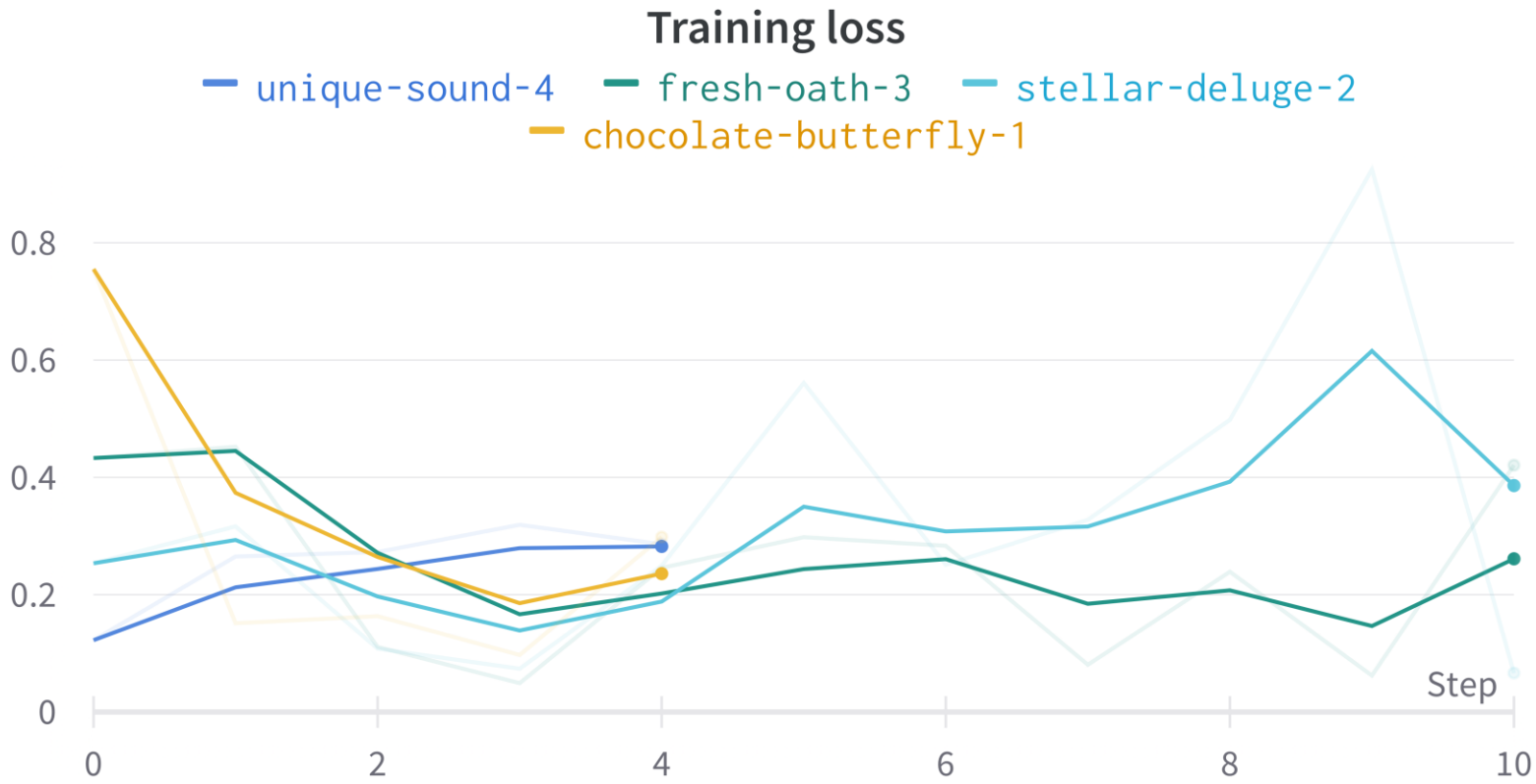
Learning rate:

- Wysokie wartości learning rate na początku przebiegu pozwalają na szybkie dopasowanie modelu do danych treningowych
- Malejące wartości wraz z upływem epok umożliwiają modelowi lepsze wpasowanie i uzyskanie lepszych wyników
- Lepsze wyniki z niższym lr, też wybrane ze względu na szybszy czas trenowania modelu, mniejsza szansa na przeuczenie się modelu



Training loss:

- Wykres ten przedstawia postęp uczenia się modelu (oś Y to wartość straty treningowej) dla przebiegów 1, 2, 3 oraz 4
- Strata treningowa powinna mieć trend malejący



Udane przebiegi:

- Ostatnie cztery przebiegi (5, 6, 7, 8) zakończyły się bez żadnych błędów. Tak jak w wypadku pierwszych czterech, parametry były zmieniane oraz ostatnie trzy (6, 7, 8) miały dodaną metrykę F1, precision, recall i accuracy).
- Batch size wybrany był stosunkowo niewielki (16/32), przy wyższych wartościach, typu 64, program się zawieszał (prawdopodobnie zbyt duże wykorzystanie pamięci)
- Argumenty w ostatnich przebiegach:

RUN 5
LR: 1e-4
EPOCHS: 1
TRAIN BATCH SIZE: 16
EVAL BATCH SIZE: 32

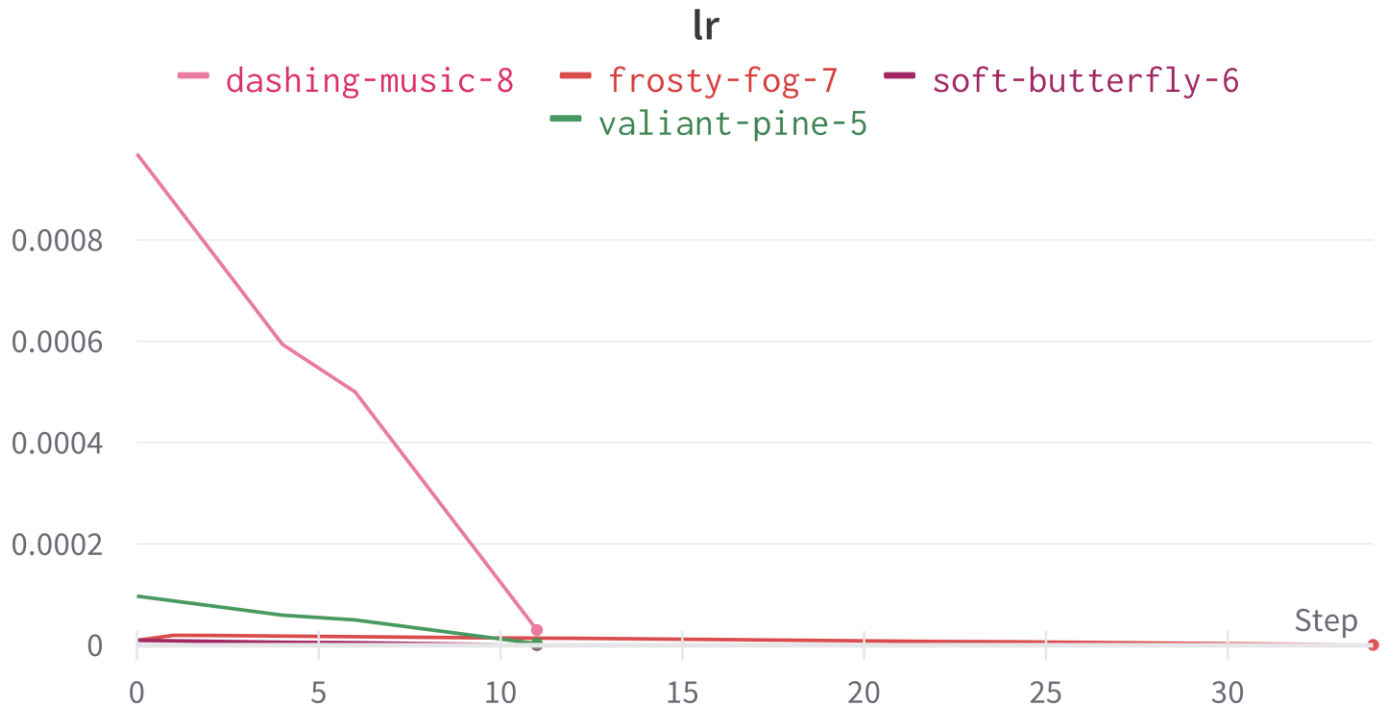
RUN 6
LR: 1e-5
EPOCHS: 2
TRAIN BATCH SIZE: 32
EVAL BATCH SIZE: 32

RUN 7
LR: 2e-5
EPOCHS: 3
TRAIN BATCH SIZE: 16
EVAL BATCH SIZE: 32

RUN 8
LR: 1e-3
EPOCHS: 2
TRAIN BATCH SIZE: 32
EVAL BATCH SIZE: 32

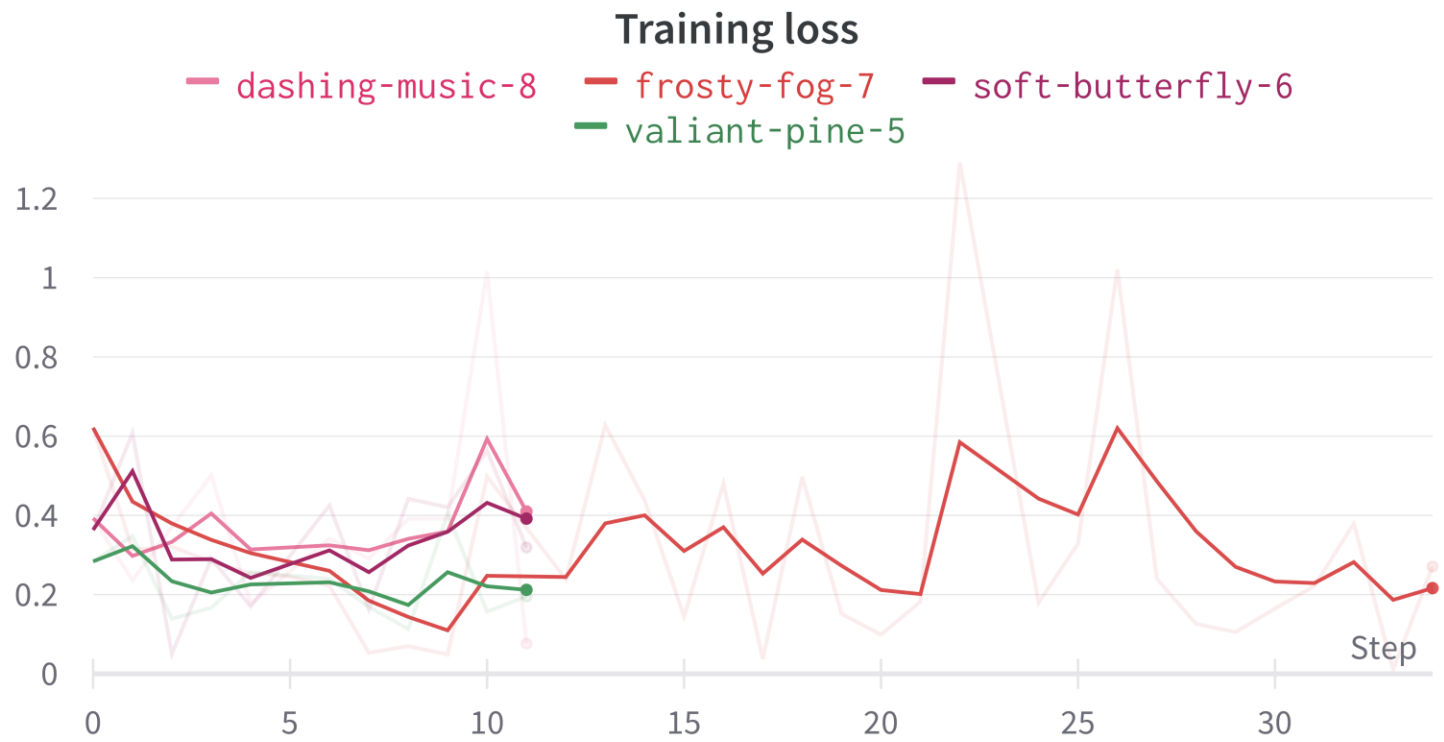
Learning rate:

- Wysokie wartości learning rate na początku przebiegu pozwalają na szybkie dopasowanie modelu do danych treningowych
- Malejące wartości wraz z upływem epok umożliwiają modelowi lepsze wpasowanie i uzyskanie lepszych wyników



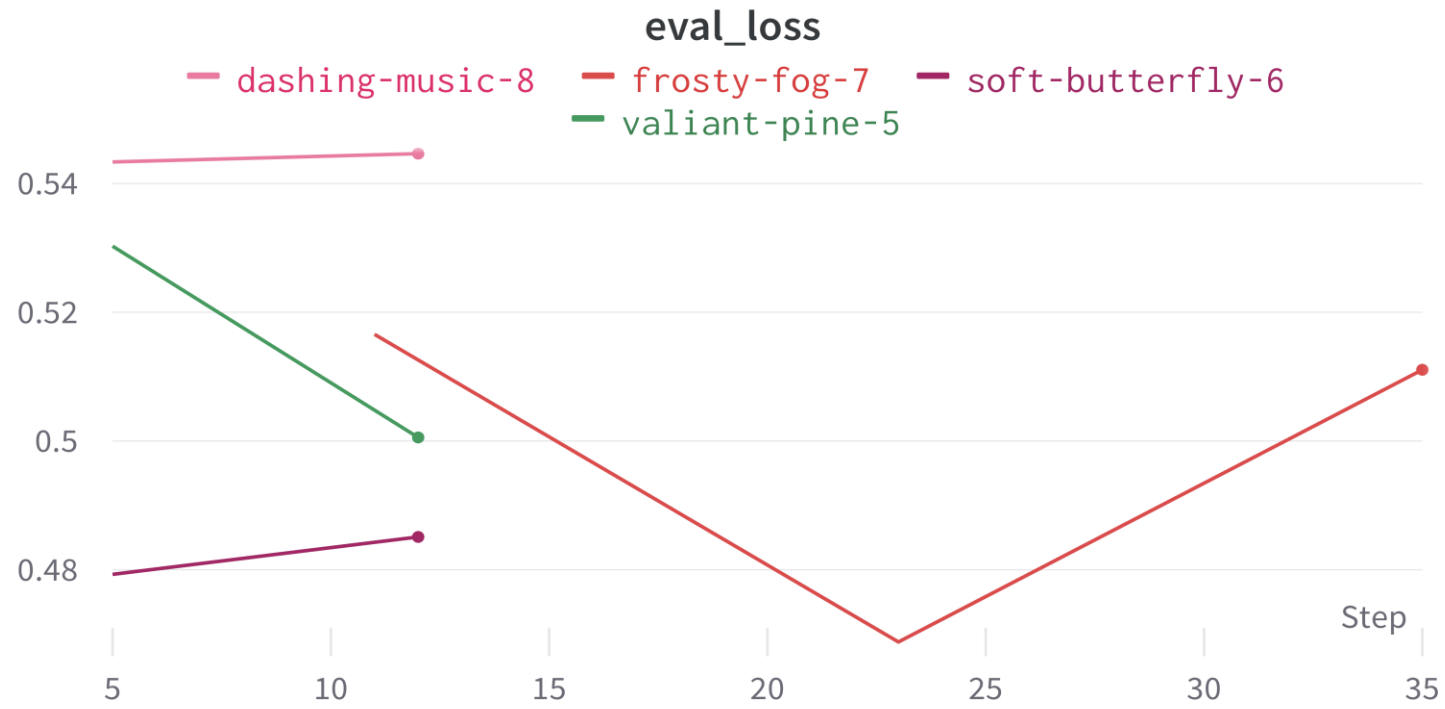
Training loss:

- Przebieg 7 ma najwięcej kroków ze względu na 3 epoki
- Strata treningowa powinna mieć trend malejący (najlepiej wypadł przebieg valiant-pine-5 oraz frosty-frog-7, ale tylko w początkowej fazie). Może to świadczyć o przeuczenie modelu i użyciu za dużej ilości epok lub zbyt wysokiego learning rate



Evaluation loss:

- Wykres ten przedstawia jak dobrze model sobie radzi z danymi z ewaluacji
- Tak jak w przypadku traning loss, strata dla ewaluacji powinna mieć trend malejący
- Najlepiej wyszło w przebiegu valiant-pine-5, który jako jedyny miał 1 epokę (reszta 2-3). Może to świadczyć o tym, że już więcej epok powoduje przeuczenie modelu (widać wzrost w przebiegu 6, 7 i 8). Przebieg frosty-frog-7 miał dobry początek lecz później odbił mocno w górę (3 epoki)

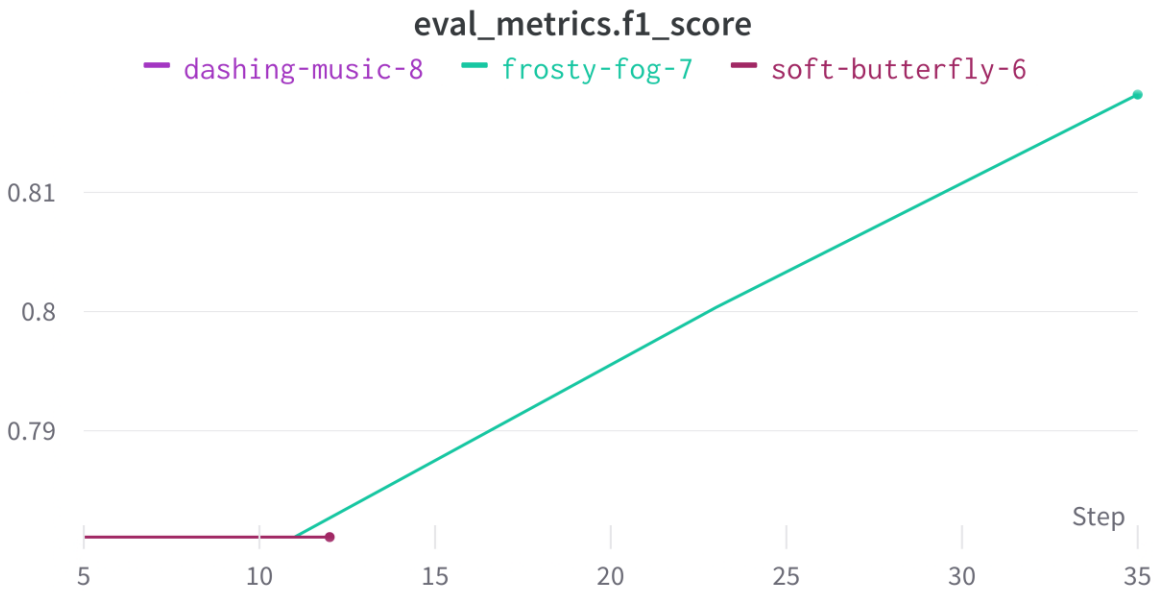


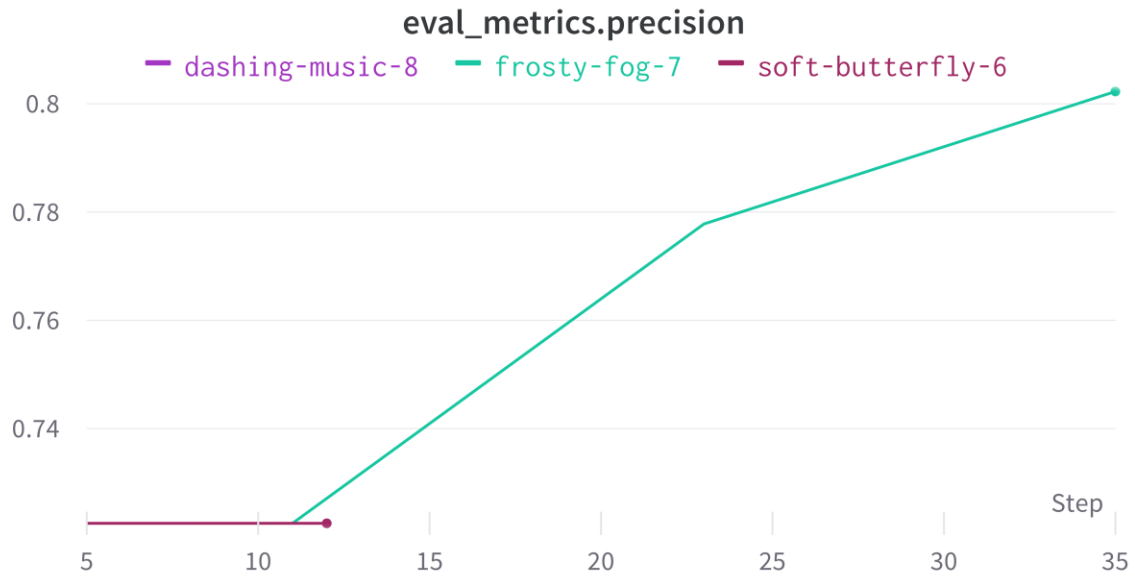
Metryki:

- Metryki dla przebiegów 6, 7 oraz 8
- Najwyższe wyniki uzyskał przebieg 7 (mimo 3 epok i wysokiego evaluation loss)
- Przy takich wartościach można powiedzieć, że model dobrze sobie radzi zarówno z przewidywaniem pozytywnych przypadków (hatespeech/cyberbullying) jak i negatywnych przypadków (wydźwięk neutralny)
- Na pewno jednak można poprawić model by wartości były bliżej 1

	F1	PRECISION	ACCURACY	RECALL
RUN 6	0.78	0.72	0.85	0.85
RUN 7	0.82	0.80	0.86	0.86
RUN 8	0.78	0.72	0.85	0.85

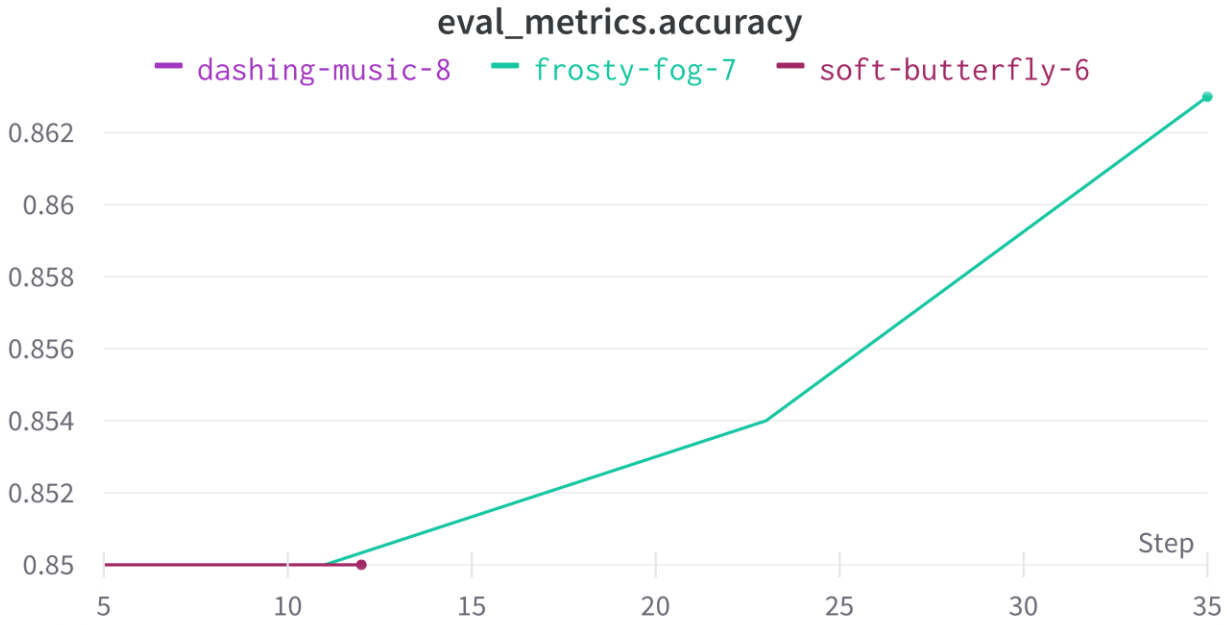
- F1 jako uśredniona precyzja i czułość
- Precyzja pokazujący jak dobrze model klasyfikuje pozytywne przypadki

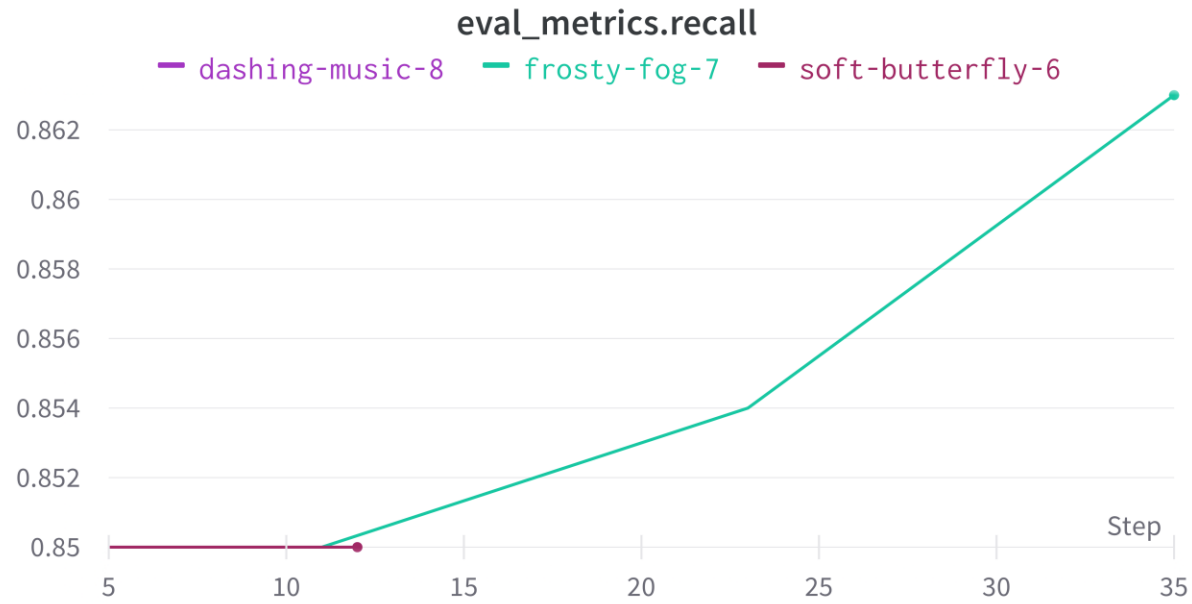




- Dokładność mierzy ogólną poprawność klasyfikacji, czyli stosunek poprawnie zaklasyfikowanych etykiet do wszystkich etykiet. Wyższa wartość accuracy wskazuje na lepszą skuteczność klasyfikacji.

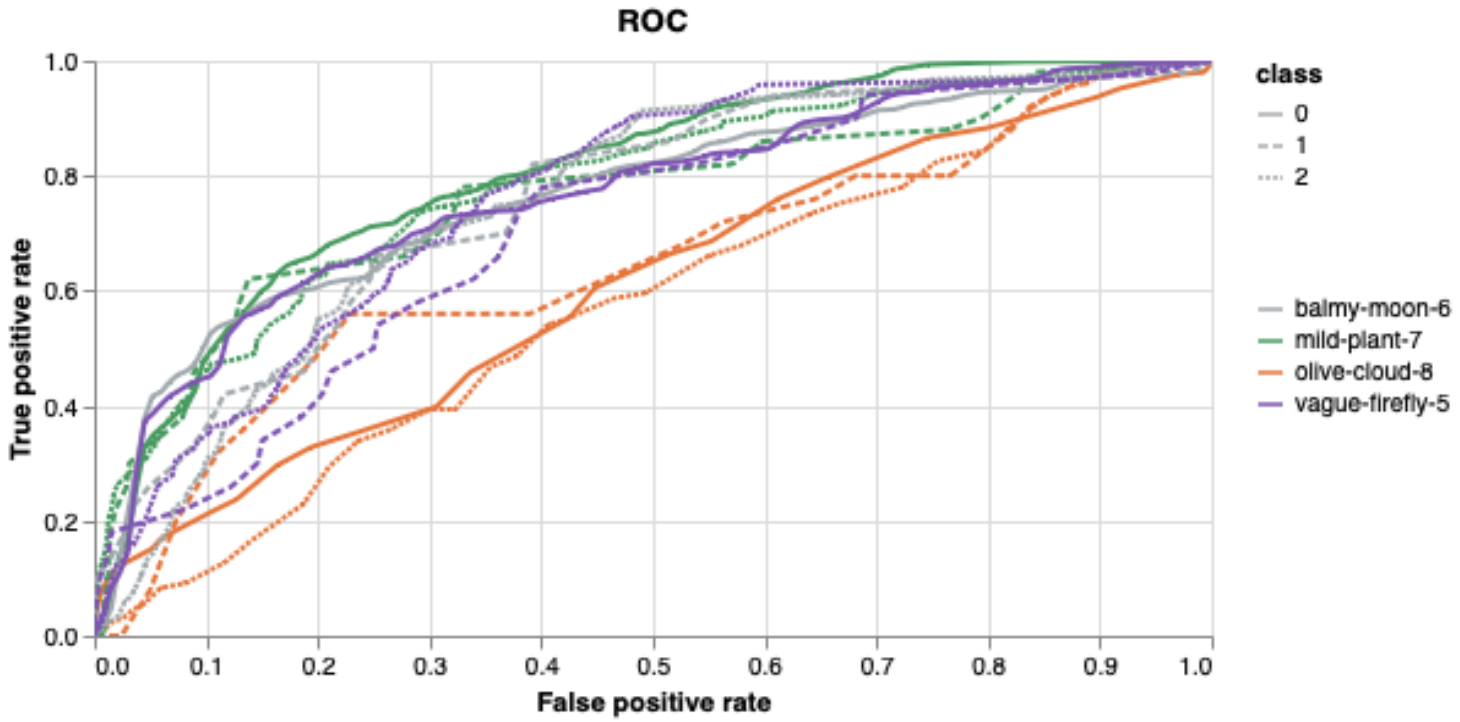
- Czułość mierzy zdolność modelu do poprawnego zidentyfikowania elementów





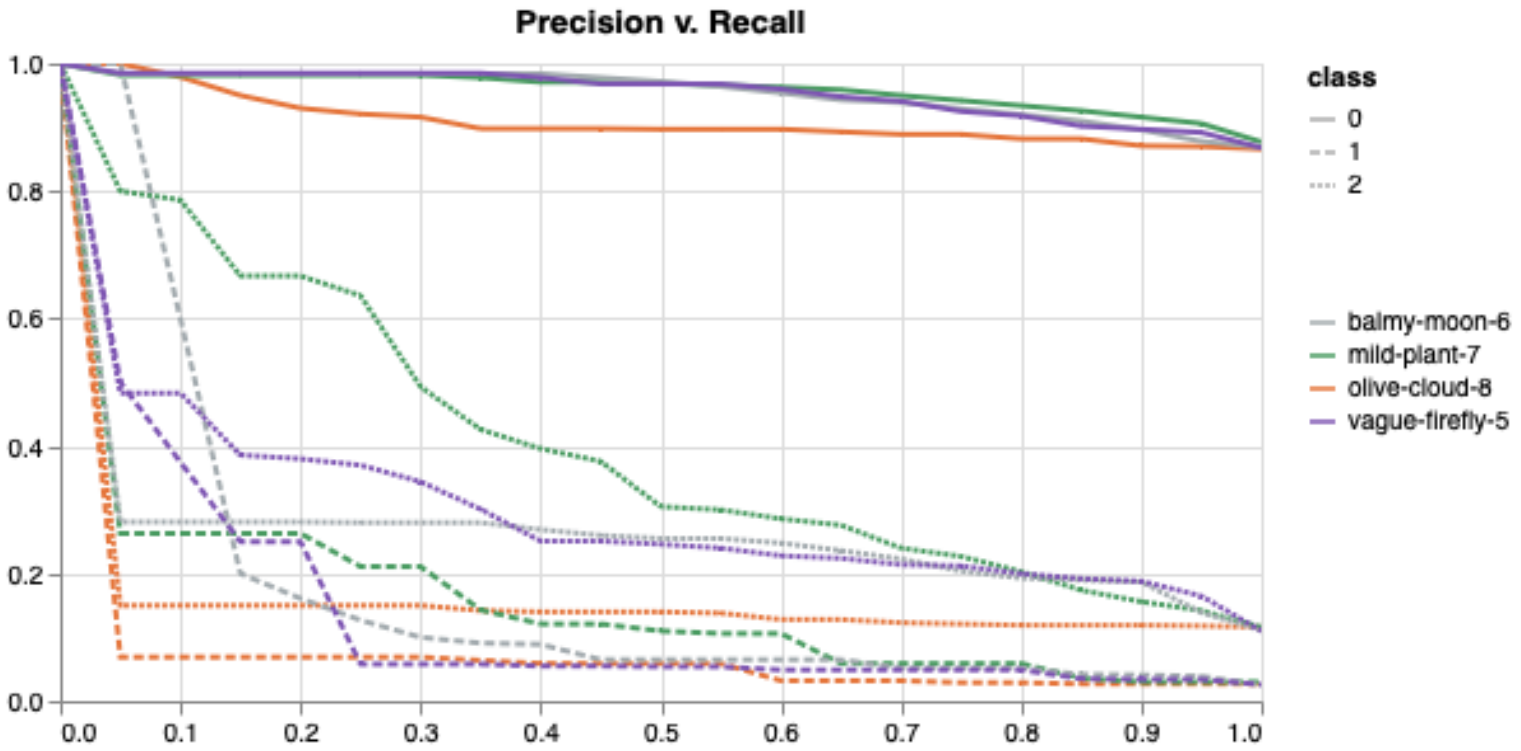
ROC:

- Wykres ROC ilustruje zdolność modelu do rozróżniania między klasami (0, 1, 2)
- Im wyżej położony jest punkt na ROC curve, tym wyższa jest wartość TPR, co oznacza lepszą zdolność klasyfikatora do identyfikowania pozytywnych próbek przy minimalnej liczbie fałszywie pozytywnych klasyfikacji
- Dobry klasyfikator powinien się łączyć od (0,0) do (1,1), co widać na poniższym wykresie głównie dla przebiegu 5 i 7. Przebieg 8 spisał się najgorzej, bardziej przecina wykres w linii prostej niż jako krzywa



Precision vs Recall:

- Precyzja (oś X) oraz czułość (oś Y).
- Idealny przypadek to punkt zbilansowany między oboma parametrami.
- Najlepiej dla klasy 1 i 2 wypadł przebieg mild-plant-7 (jest najbardziej zbilansowany)
- Dla klasy 0 jest bardzo wysoka czułość i precyzja
- Dla klasy 1 i 2 jest wysoka jedynie precyzja, czyli mamy minimalizację fałszywie pozytywnych klasyfikacji



TF-IDF

- Użycie TF-IDF oraz klasyfikatora (logistyczna regresja) za pomocą Sklearn służyło porównaniu wyników otrzymanych z wcześniejszego trenowania modelu
- Kroki:
  - Przekształcenie danych treningowych i ewaluacyjnych na wektory za pomocą TF-IDF
  - Trening klasyfikatora na danych treningowych i odpowiadających etykietach
  - Dokonanie predykcji na danych ewaluacyjnych i porównanie danych z predykcji z prawdziwymi etykietami
  - Otrzymane wyniki były porównywalne do otrzymanych z modelu BERT

```
F1: 0.7892922844235707
Accuracy: 0.853
Precision: 0.7783611670020122
Recall: 0.853
```

	F1	PRECISION	ACCURACY	RECALL
RUN 6	0.78	0.72	0.85	0.85
RUN 7	0.82	0.80	0.86	0.86
RUN 8	0.78	0.72	0.85	0.85
TF-IDF	**0.79**	**0.78**	**0.85**	**0.85**

Podsumowanie

Wykonanie projektu pozwoliło na zrozumienie problemu jakim jest rozpoznawanie mowy nienawiści i cyberprzemocy w tekstach pochodzących z Internetu. Zastosowane metody mają zarówno zalety, jak i wady. Uzyskane wyniki nie są w 100% satysfakcjonujące. Znaczącym problemem był długi czas oczekiwania na wynik treningu modelu (około 1,5h na epokę), ponieważ błędy można było poprawiać dopiero po zakończeniu analizy, a to bardzo utrudniało pracę. Ogromny wpływ na rezultaty miał również nierównomierny rozkład danych i większość tych o nacechowaniu neutralnym. Pomóc mógłby oversampling lub undersampling, ale niestety i te sposoby mogłyby przynieść problemy w postaci przetrenowania modeli lub utraty danych z przeważającej klasy.

Inżynieria Lingwistyczna

Projekt: Wykrywanie mowy nienawiści i cyberprzemocy za pomocą BERT

Michalina Kwolik s23922
Gabriela Olszewska s28652