Efektywność Modeli Głębokich i Płaskich w Klasyfikacji Tekstów Polskich Autorów

Krystyna Waniová

1 Wstęp

Praca porównuje wyniki dwóch głębokich sieci neuronowych (jednej z warstwami liniowymi oraz drugiej z warstwami konwolucyjnymi) z wynikami klasycznych modeli maszynowego uczenia (Random Forest i Logistic Regression) w kontekście klasyfikacji tekstów polskich autorów.

2 Dane

Dane w projekcie pochodzą ze strony wolnelektury.pl. Znajdują się w folderze data. Do klasyfikacji wykorzystano teksty dzieł autorstwa Adama Mickiewicza, Juliusza Słowackiego, Władysława Reymonta oraz Henryka Sienkiewicza.

3 Wykorzystane technologie

Projekt jest napisany w języku Python. Wykorzystane narzędzia:

- **Nltk** - narzędzie do przetwarzania języka naturalnego (NLP). Wykorzystane w projekcie do segmentacji oraz preprocessingu danych.
- **Pytorch** - biblioteka do uczenia maszynowego.
- **Scikit-learn (sklearn)** - biblioteka do uczenia maszynowego. Oferuje zaimplementowane gotowe modele oraz funkcje do analizy danych.
- **Pandas, Numpy**

4 Architektura projektu

4.1 Przygotowanie danych

W pierwszej części projektu przetwarzam dane na których uczone są później modele. Tekst każdego z autorów jest przetworzony za pomocą funkcji tokenize_and_normalize w następujący sposób:

- **Tokenizacja** - Zdania są poddawane tokenizacji przy użyciu polskiego tokenizera z biblioteki nltk.
- **Usuwanie znaków nie ASCII**
- **Zastępowanie liczb słowami**
- **Sprowadzanie tekstu do małych liter**
- **Usuwanie znaków interpunkcyjnych.**
- **Usuwanie stopwords** - znajdujących się w pliku polish_stopwords.txt

```
def normalize(words):
    """
    Normalize words
    """
    words = remove_non_ascii(words)
    words = replace_numbers(words)
    words = to_lowercase(words)
    words = remove_punctuation(words)
    words = remove_stopwords(words)
    return words
```

Figure 1: Funkcja proprocessingu tekstu

Kolejnym etapem jest przypisanie etykiet do danych, a następnie utworzenie dataframe'u, w którym znajdują się dane od wszystkich autorów w losowej kolejności.

	text	label
0	z nas wszystkich kiedys beda takie trupy	1
1	znow bysna promien niby na oboku w obraczce te...	2
2	gdyby te dary gdy nie przerazona mysl na co by...	1
3	niech twoja rodzina westchnie za mna do boga	1
4	modziez posza do lasu bawic sie strzelbami a s...	0

Następuje podział danych na dane testowe, walidacyjne oraz treningowe. W tym celu wykorzystana jest funkcja z biblioteki sklearn train_test_split a następnie tworzone są dataloadery do uczenia głębokich modeli, które operują na batchach danych.

Następnie użyty został TfidfVectorizer do stworzenia reprezentacji tekstu w postaci wektorów liczbowych.

4.2 Przygotowanie modeli

W projekcie porównuję kilka sieci:

- **Net** - prosta sieć z warstwami liniowymi, funkcją aktywacji Relu, wykorzystującą dropout oraz batch norm. Na końcu przejścia forward znajduje się Softmax do określenia prawdopodobieństw w klasyfikacji wieloklasowej.
- **ConvolutionalNet** - sieć zawierająca warstwy konwolucyjne, funkcję aktywacji Relu, dropout, batch norm oraz softmax do klasyfikacji wieloklasowej.
- **LogisticRegression** - wykorzystałam implementację z biblioteki sklearn. Logistic regression to technika statystyczna wykorzystywana do przewidywania prawdopodobieństw przynależności danych do jednej z klas.
- **RandomForest** - wykorzystałam implementację z biblioteki sklearn. Random Forest wykorzystuje zbiór losowo generowanych drzew decyzyjnych do prognozowania przynależności danych do poszczególnych klas.

4.3 Przebieg uczenia

Modele LogisticRegression oraz RandomForest uczone są za pomocą metody fit() do danych treningowych. Dla modeli głębokich stworzona została pętla ucząca w ktorej wyświetlam przebieg uczenia modelu. W uczeniu wykorzystuję CrossEntropyLoss jako funkcję kosztu oraz optimizer Adam.

Dla modelu liniowego optymalny learning rate który udało mi się znaleźć wynosi 0.00003. Dla modelu konvolucyjnego użyłam leraning rate 0.00001.

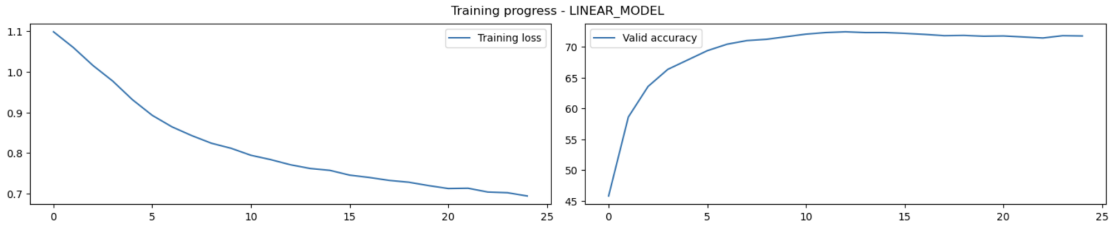

Figure 2: Przebieg uczenia modelu liniowego - Net - 25 epok

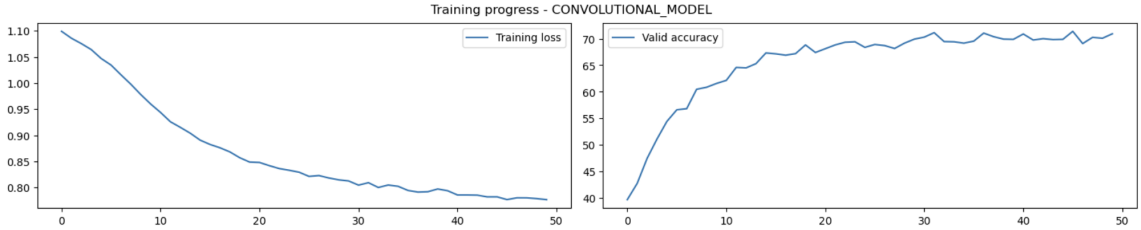

Figure 3: Przebieg uczenia modelu konwolucyjnego - ConvolutionalNet - 50 epok

4.4 Testowanie modelu

Do testowania modeli służy funkcja test_model, w której obliczam całkowite accuracy oraz roc_auc modelu. Następnie podliczam też accuracy modelu dla każdej z klas (czyli dla każdego autora).

Następnie wyniki dla poszczególnych modeli zapisywane są w folderze results.

4.5 Wyniki

Wyniki testów dla wszystkich modeli są zaprezentowane poniżej:

```
Accuracy for class: 0 - mickiewicz is 56.4 %
Accuracy for class: 1 - slowacki is 73.5 %
Accuracy for class: 2 - sienkiewicz is 71.6 %
              precision    recall  f1-score   support

           0      0.728     0.564     0.636       815
           1      0.596     0.735     0.658       706
           2      0.702     0.716     0.709       860

    accuracy                          0.670      2381
   macro avg      0.675     0.672     0.668      2381
weighted avg      0.679     0.670     0.669      2381

Overall metrics:
Accuracy: 66.99%
Roc auc score: 0.853
```

Figure 4: Wyniki testów dla modelu Random Forest

```
Accuracy for class: 0 - mickiewicz is 66.9 %
Accuracy for class: 1 - slowacki is 75.1 %
Accuracy for class: 2 - sienkiewicz is 75.9 %
              precision    recall  f1-score   support

           0      0.735     0.669     0.700       815
           1      0.680     0.751     0.714       706
           2      0.759     0.759     0.759       860

    accuracy                          0.726      2381
   macro avg      0.725     0.726     0.724      2381
weighted avg      0.727     0.726     0.726      2381

Overall metrics:
Accuracy: 72.57%
Roc auc score: 0.883
```

Figure 5: Wyniki testów dla modelu Logistic Regression

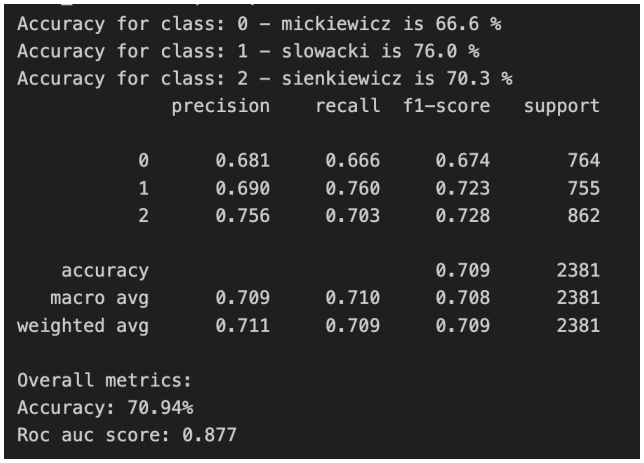

Figure 6: Wyniki testów dla modelu lioniowego Net

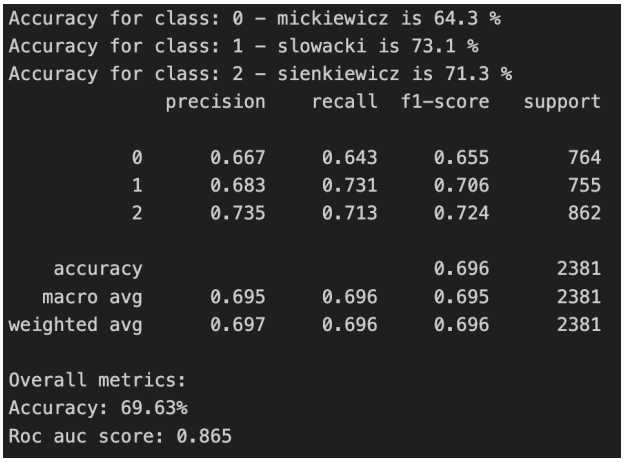

Figure 7: Wyniki testów dla modelu konwolucyjnego ConvolutionalNet

	model	accuracy	roc_auc
3	linear_model	70.936581	0.876736
1	convolutional_model	69.634607	0.864504
0	logistic_regression	68.247220	0.889272
2	random_forest	60.954872	0.844272

Figure 8: Wyniki porównawcze dla wszystkich modeli i wszystkich 4 autorów

	model	accuracy	roc_auc
0	logistic_regression	71.314574	0.875454
3	linear_model	70.936581	0.876736
1	convolutional_model	69.634607	0.864504
2	random_forest	65.938681	0.843406

Figure 9: Wyniki porównawcze dla wszystkich modeli i 3 autorów (bez Reymonta)

Najlepiej radziły sobie modele LogisticRegression oraz model liniowy Net. Model z warstwami konwolucyjnym oraz RandomForest zazwyczaj dawały gorsze wyniki.