Identifying and Categorizing Offensive Language in Social Media

Kacper Gocłowski, Krzysztof Filipów

11 czerwca 2021

1 Przygotowanie oraz preprocessing danych

1.1 Wczytanie danych

Dane zostały wczytane z plików tsv i csv do DataFrame'ów Pandasa. Wyodrębnione zostały tweety i odpowiadające im labele do subtasków A i B oraz zestawy tweetów testowych, również z odpowiednimy labelami.

1.2 Preprocessing danych

Tweety w tym momencie stanowią ciągi wyrazów. Należy je przekształcić na ciągi liczb za pomocą encodera. W tym celu zastosowano TextVectorization z Kerasa z domyślną standardyzacją (lower case i bez interpunkcji).

Później prezentuje się odtworzenie tweetów z postaci liczbowej do słów w celu weryfikacji poprawności. Dokładniej przekonania się ile słów stało się Unknownami przy podanym rozmiarze słownika.

To samo wykonano dla danych do subtaska B.

Następnie ręczne przetworzenie labeli na liczby, 'NOT' na 0, 'OFF' na 1. Podobnie dla subtaska B: 'TIN' na 1 i 'UNT' na 0.

To kończy wstępne przygotowanie danych.

2 Architektura i trening sieci neuronowej

2.1 Budowa sieci

Nadszedł czas na budowę sieci. Do obu subtasków sieć ma taką samą strukturę. Wykorzystano Keras Sequential. Pierwszą warstwą jest utworzony uprzednio encoder, który przekształca tekst na liczby. Następnie warstwa Embedding, która przekształca słowa-liczby na wektory. Po czym następuje warstwa rekurencyjna LSTM oraz jedna zwykła warstwa Dense z funkcją aktywacji relu. Ostatnia warstwa jest z funkcją aktywacji sigmoidalną. Dodano dropout 0.5 do każdej z warstw ukrytych.

2.2 Parametry uczenia

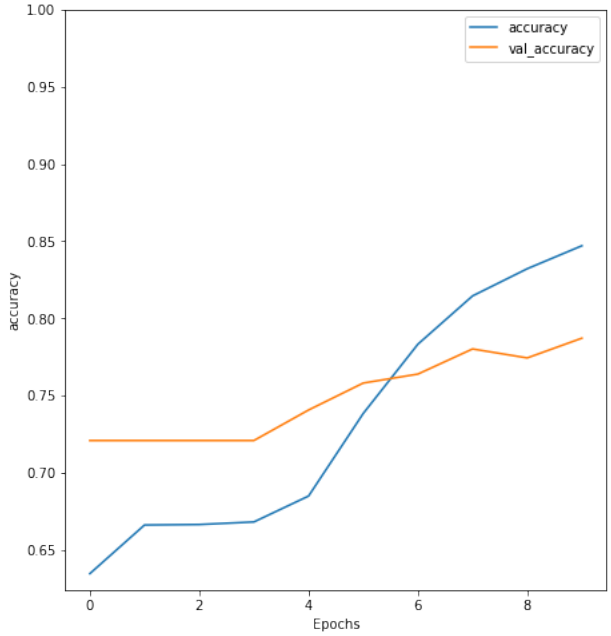
Celem jest przetworzenie tweeta na liczbę między 0 i 1, w zależności czy jest on Offensive -> bliżej 1, czy Not Offensive -> bliżej 0; binarna klasyfikacja. Z tego względu loss liczony jest jako BinaryCrossentropy. Zastosowano optymizer Adam z learning ratem 0.0001.

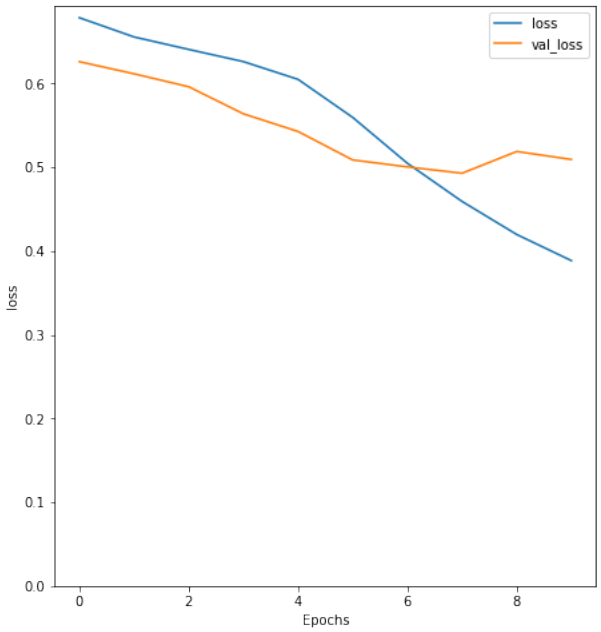
2.3 Uczenie

Następnie wykonywane jest uczenie sieci. Po czym następuje powstanie i nauka sieci dla subtaskaB.

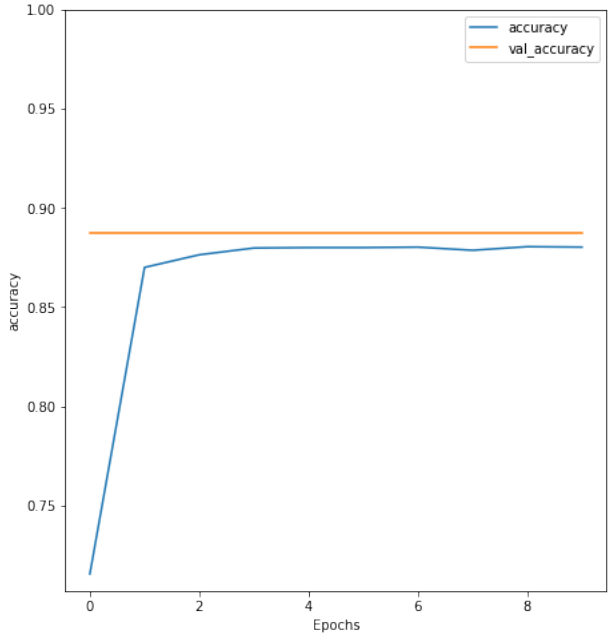
2.4 Wyniki uczenia

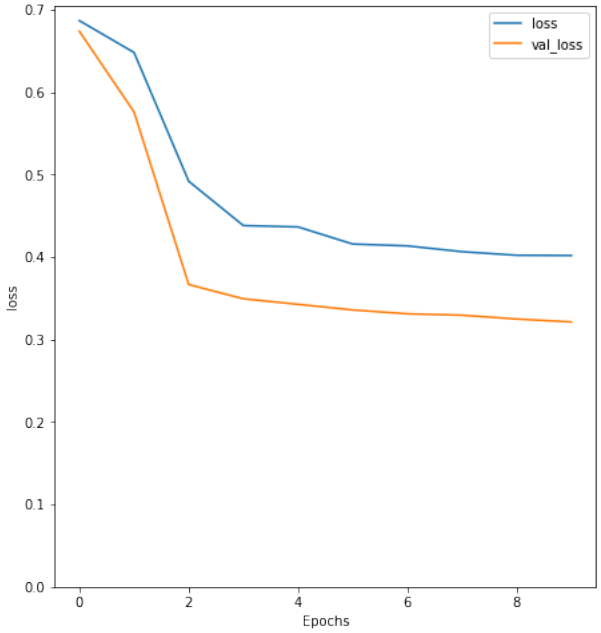
Poniżej wypisanych jest kilka przykładów odpowiedzi sieci z wzorcowymi labelami.
Następnie wyliczane jest precision, recall i f1 dla obu modeli.
Na koniec prezentują się wykresy z uczenia funkcji straty (loss) i celnosci (accuracy) kolejnych epok.
Dla modeluA:


accuracy
val_accuracy
accuracy
Epochs


loss
val_loss
loss
Epochs

Dla modeluB:


accuracy
val_accuracy
accuracy
Epochs


loss
val_loss
loss
Epochs

3 Podsumowanie

Takie rozwiązanie pozwala już od początku uczenia dość trafnie rozpoznawać tweety. Zabawa z ilością warstw, ich rozmiarem oraz parametrami uczenia niewiele wpływa na wynik. Wpływa za to znacznie na długość uczenia.
Sieć A i jej wykresy uczenia wyglądają standardowo. Sprawa wygląda inaczej w przypadku B. Tutaj uczenie niewiele daje i to przy bardzo różnych parametrach (udało się wyczerpać limit korzystania z GPU na collabie testując). Dość łatwo tu o overfitting, gdzie już po kilku epokach accuracy treningowe jest >0.9, a testowe tylko maleje.
W subtasku B sieć ma problem z nauczeniem się "0". Na co wpływ ma między innymi rozkład danych testowych, gdzie są prawie same "1".

3.1 Miary

Biorąc pod uwagę wyliczone miary, w zadaniu A: lepiej klasyfikowana jest klasa 0 -> Not Offensive, jednak z klasą 1 też nie ma tragedii. Za to w zadaniu B klasa 1 -> Targeted jest dobrze rozpoznawana, a klasa 0 bardzo słabo.

3.2 Wnioski na przyszłość

Patrząc w przyszłość, należałoby rozważyć inne możliwości wstępnego przetwarzania tweetów-słów.