Projekt UMA Kacper Dudzic

06-DUMAUI0 2022/SL

Cel projektu

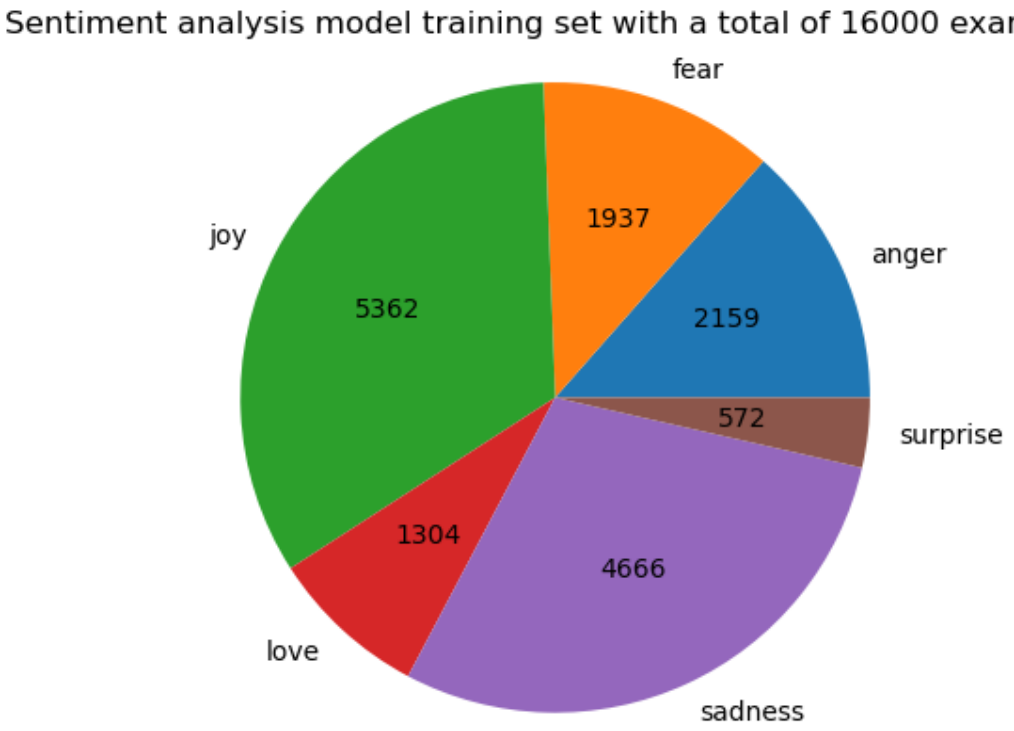
Celem projektu było stworzenie modelu, który potrafi zidentyfikować wydźwięk emocjonalny w krótkich wypowiedziach anglojęzycznych na podstawie ich treści. Model ten realizuje więc klasyczne zadanie analizy wydźwięku, tutaj w ujęciu wieloklasowym.

Dane

Dane pochodzą z publicznie dostępnego zestawu danych „Emotions dataset for NLP" zamieszczonego na platformie Kaggle przez użytkownika praveengovi.

(https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp)

Zawartość zestawu to dwa pliki – odpowiednio dane treningowe i testowe – złożone z krótkich tekstów w języku angielskim opatrzonych etykietami opisującymi ich wydźwięk emocjonalny w ramach 6 uwzględnianych emocji: *joy*, *fear*, *anger*, *surprise*, *love* i *sadness*. Dane treningowe liczą 16000 przykładów, a testowe – 2000. Z racji charakteru danych nie było potrzeby odrzucania żadnych przykładów.



^ Wykres kołowy generowany przez skrypt run_models.py.

Modele

W projekcie porównano działanie 3 modeli:

- Naiwny klasyfikator bayesowki z domyślnymi ustawieniami w ramach implementacji z biblioteki Scikit-learn. Do wektoryzacji tekstów użyto wektorów tf-idf. Model ten, jako proste i obliczeniowo tanie rozwiązanie, posłużył jako pewien „baseline" dla pozostałych modeli.
- Sieć neuronowa typu LSTM zaimplementowana za pomocą biblioteki Keras/Tensorflow. Teksty zwektoryzowane domyślną klasą Tokenizer i warstwą Embedding. Warstwa LSTM dwukierunkowa. Pełna architektura sieci wyświetlana w ramach skryptu run_models.py.
- Sieć neuronowa typu Transformer wykorzystująca pretrenowany model BERT zaimplementowana za pomocą bibliotek Transformers i Keras/Tensorflow. Wektoryzacja tekstów klasą AutoTokenizer wykorzystującą BERT-a. Pełna architektura sieci wyświetlana w ramach skryptu run_models.py.

Ewaluacja

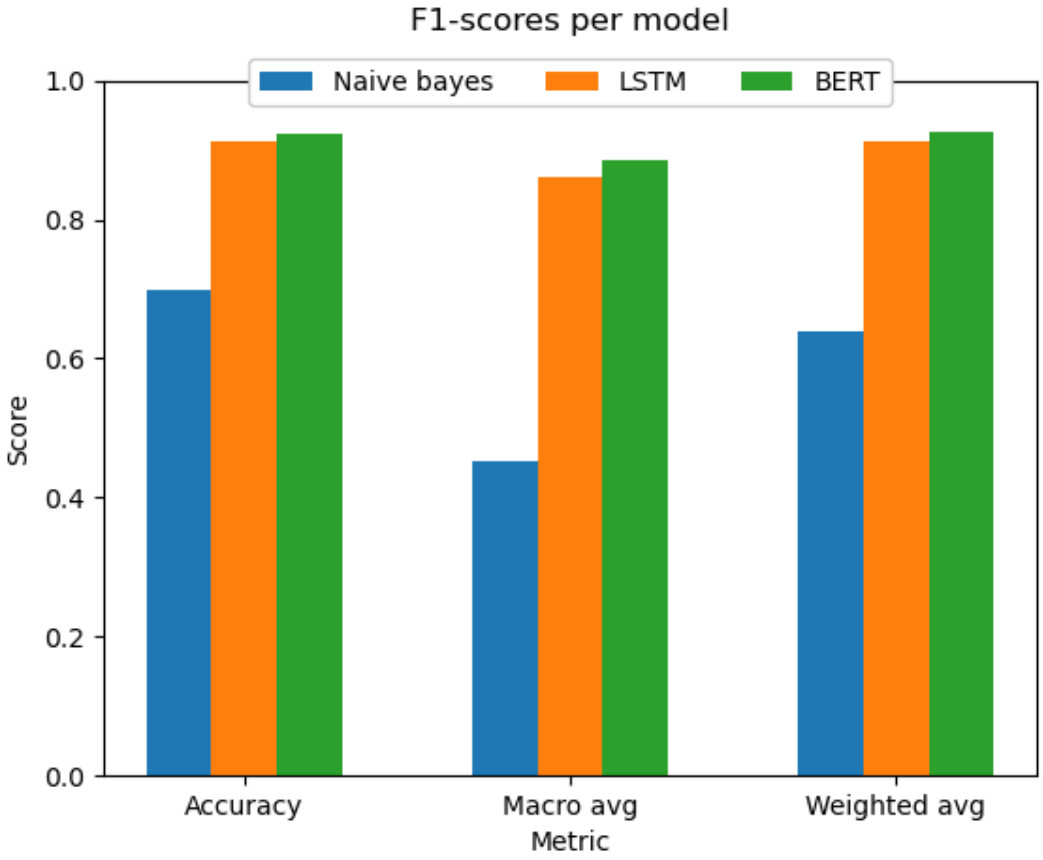
Do ewaluacji wykorzystano metryki *accuracy, precision, recall* i *F1-score*. Wyniki ewaluacji przedstawia poniższa tabelka:

Model	Accuracy	Precision*	Recall*	F1-score*
Naiwny klasyfikator bayesowski	70%	87% 77%	44% 70%	45% 64%
Sieć neuronowa typu LSTM	91%	89% 92%	85% 91%	86% 91%
Sieć neuronowa typu Transformer (BERT)	92%	87% 93%	91% 92%	89% 92%

*Górna wartość w komórkach precision/recall/F1-score to *macro average*, a dolna to *weighted average*.

Wnioski

Najlepszym modelem okazał się – przewidywalnie – model z BERT-em, będący najnowszą i najefektywniejszą architekturą spośród 3 wybranych. Osiągnął on świetne wyniki dla wszystkich metryk, rzędu ok. 90%. Mimo tego, model wykorzystujący sieć LSTM okazał się jedynie odrobinę gorszy, odstając o kilka punktów procentowych na przestrzeni wszystkich metryk oprócz *micro average* dla *precision*, które wyszło 2% lepiej. LSTM jest naturalnie kolejną z metod uznawanych za bardzo efektywne w zakresie problemów dotyczących danych tekstowych. Niewielka różnica pomiędzy tymi dwoma modelami może wynikać z, jak się okazuje, małej trudności samego zadania, do którego zostały wykorzystane – w obu przypadkach postawiony przed nimi problem został zasadniczo „rozwiązany". Przewidywalnie spośród 3 modeli najgorzej wypadł naiwny klasyfikator bayesowski, ale mimo wszystko wyniki przez niego osiągane też są dosyć przyzwoite – całokształt psują jedynie niskie wartości *macro* metryk *recall* i *F1-score*. Biorąc pod uwagę, że w przeciwieństwie do pozostałych modeli wyniki zwracane są przez niego zasadniczo od razu, mógłby on z powodzeniem okazać się wystarczający, a nawet pożądany w przypadku pewnych prostszych praktycznych zastosowań analizy wydźwięku.



^ Wykres słupkowy generowany przez skrypt run_models.py.