# Predykcja popularnych emoji za pomocą wybranych modeli głębokich

## 1 Opis problemu

Projekt dotyczy predykcji popularnych emoji (piktogramy używane często w mediach społecznościowych w celu wyrażania emoji) dla danego fragmentu tekstu. Zbiór danych pozyskany został z serwisu Twitter. Zawiera on ponad pół miliona tweetów. W projekcie przebadany został wpływ różnych architektur sieci głębokich i ich parametrów na wyniki predykcji.

## 2 Stan literaturowy

Inspiracją do podjęcia tematu był konkurs SemEval z 2018 roku – Multilingual Emoji Prediction (1). Wcześniej ukazała się również praca autorów konkursu, którzy pokazali, że model obliczeniowy potrafi lepiej niż człowiek odczytać ukryte znaczenie emoji (2). Do predykcji emoji została tam zastosowana sieć LSTM. Również po zakończeniu konkursu powstały ciekawe prace na temat klasyfikacji emoji: (3) (zwycięzcy konkursu, w pracy pokazali jak za pomocą SVM osiągnęli lepsze rezultaty niż stosując sieci Bi-LSTM), (4) (zastosowanie Bi-LSTM z mechanizmem uwagi oraz pretrenowanych wektorów word2vec), (5) (zastosowanie Gradient Boosting Regression Tree z Bi-LSTM). Przedstawione przykłady pozwalają stwierdzić, że jednym z najpopularniejszych rozwiązań problemu jest zastosowanie wariantu sieci LSTM (bardziej ogólnie RNN). Nie jest to jednak jedyne podejście w klasyfikacji tekstu. Innym przykładem może być zastosowanie sieci CNN jak to zostało pokazane tutaj (6), (7). Coraz popularniejsze są również rozwiązania oparte o modele typu transformer. Przykładem może być model BERT (8). Zastosowanie modelu BERT do klasyfikacji tekstów zostało omówione tutaj (9).

## 3 Wizja rozwiązania

Po zapoznaniu się ze stanem literaturowym pojawiła się następująca wizja rozwiązania. Stworzenie i porównanie trzech modeli sieci głębokich: opartego na sieci RNN, CNN oraz dostosowanie modelu BERT wykorzystanych do klasyfikacji 45 emoji oraz wyznaczonych grup.

Do rozwiązania zadania zostaną wykorzystane dane z serwisu Twitter, które zostaną oczyszczone oraz odpowiednio podzielone dla najpopularniejszych 45 emoji. Utworzonych zostanie również sześć grup podobnych emoji. Dane dla klasyfikatorów reprezentowane będą w postaci umownych indeksów słów, które będą stanowiły wejście do uczonych warstw embeddingu.

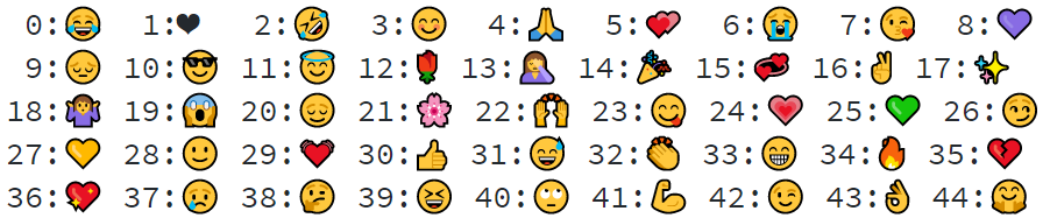
Jako przykład sieci CNN zostanie wykorzystana sieć, w której warstwy konwolucyjne są zestawione równolegle, a rozmiary filtrów są różne. Taki zabieg ma emulować działanie modeli skip-gram. Sieci rekurencyjne prezentować będą modele z warstwą Bi-LSTM oraz Bi-GRU, ponieważ stanowiły one częste rozwiązanie problemu w literaturze. Wykorzystany zostanie również wcześniej przetrenowany model BERT w połączeniu z warstwami gęstymi dla dostosowania do zadania klasyfikacji.

Eksperymentom zostaną poddane takie parametry jak liczba neuronów w warstwie, rozmiar embeddingu, prawdopodobieństwo dropoutu, czy użyty optymalizator oraz współczynnik uczenia się.

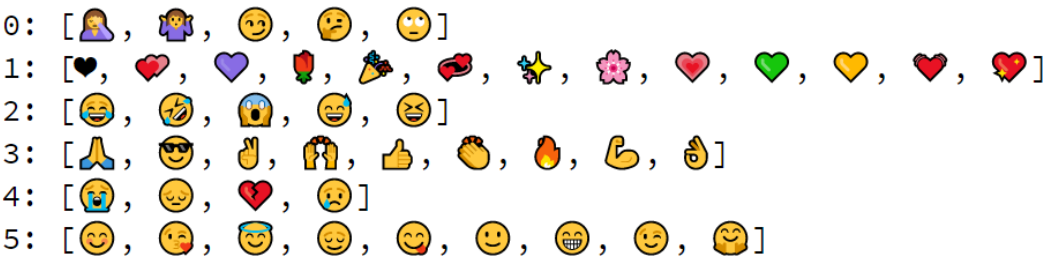
## 4 Dane

Dla zadania klasyfikacji zebranych zostało nieco ponad pół miliona tweetów. Zostały one poddane procesowi czyszczenia, w którym zostały usunięte np. linki, maile, oznaczenia innych użytkowników, liczby, daty, itd. Następnie z tekstów zostały wydzielone fragmenty zakończone pojedynczym emoji o długości co najmniej trzech słów. W wyniku tych operacji ostatecznie powstał zbiór zawierający 363945 przykłady, które zostały podzielone na zbiory treningowy (80%) i testowy (20%). Do sprawdzania parametrów w trakcie uczenia ze zbioru treningowego został dodatkowo wydzielony zbiór walidacyjny.

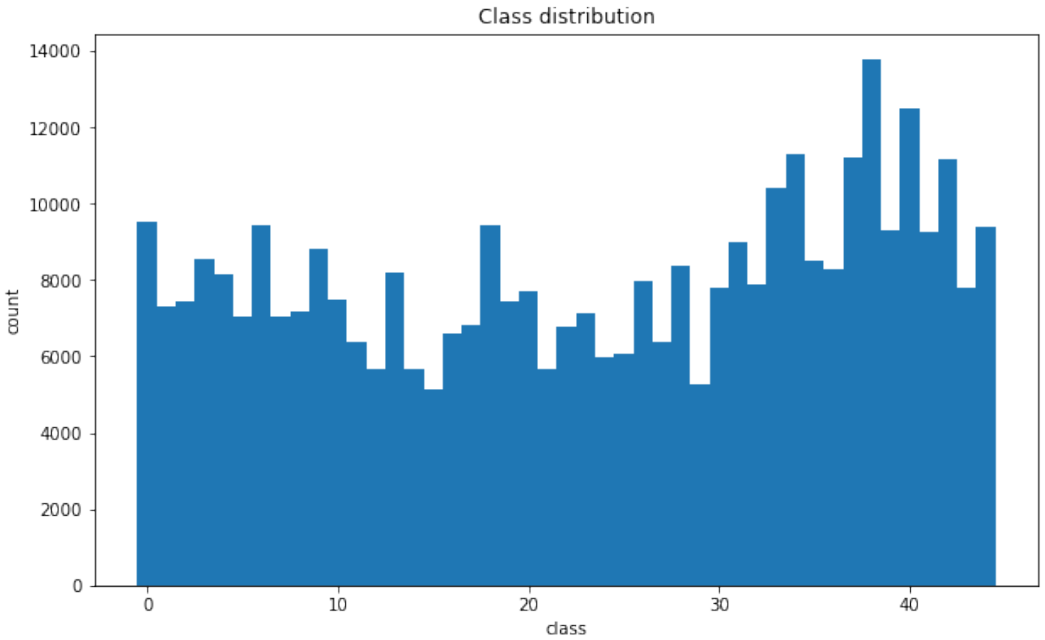
Zbiór został oetykietowany na dwa sposoby. Pierwszym z nich było przyporządkowanie jednej z 45 wybranych emoji. Drugim, jednej z 6 grup na podstawie tego do której grupy należało emoji przyporządkowane do tekstu. Poniżej zostały przedstawione emoji wraz z numerami klas oraz grupy (rysunki 1, 2) oraz rozkład klas w zbiorze danych z pojedynczymi emoji (3) oraz grupami (4).



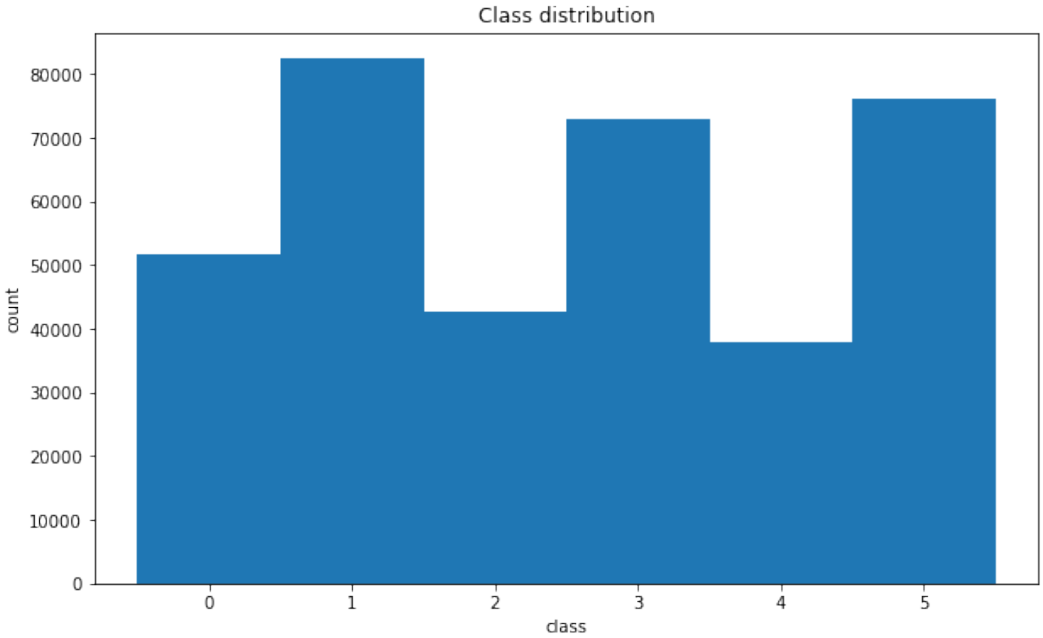

Rysunek 1: Lista emoji.




Rysunek 2: Grupy podobnych emoji.




Rysunek 3: Rozkład klas emoji.




Rysunek 4: Rozkład klas grup emoji.

## 5 CNN

Wykorzystana w zadaniu sieć CNN składa się z warstwy embeddingu, trzech warstw konwolucyjnych (1D) ułożonych równolegle o rozmiarach filtrów 3, 4 oraz 5 (z funkcją aktywacji ReLU), po których następują warstwy max-poolingu o rozmiarze takim, aby wynikowo otrzymać pole długości jeden. Następnie trzy równoległe warstwy są łączone w jedną oraz spłaszczane i podawane na wejście do warstwy w pełni połączonej z funkcją aktywacji ReLU. Na końcu sieci pojawia się warstwa dropout oraz ostatnia warstwa w pełni połączona z funkcją aktywacji softmax.

W eksperymencie przebadane zostały przebadane takie parametry jak rozmiar embeddingu, liczba neuronów w warstwach konwolucyjnych, prawdopodobieństwo dropoutu oraz dwa algorytmy optymalizacyjne – Adam oraz SGD z domyślnymi parametrami. Dla każdej z prób rozmiar batcha wynosił 500, a liczba epok 10. Wyniki przedstawiono poniżej (tabele 5, 6).

Na uzyskane wyniki znaczący wpływ miał rozmiar warstwy embeddingu. Próby z większym rozmiarem (300) uzyskiwały lepsze rezultaty niż te z mniejszym (100). Minimalnie wpłynęła również wartość dropoutu – lepiej dla 0.25 niż 0.5 oraz liczba neuronów w warstwach konwolucyjnych – lepiej 128 niż 64. Bardzo słabe rezultaty osiągnięto za pomocą algorytmu SGD (model praktycznie się nie uczy), dlatego w następnych eksperymentach większy nacisk kładziony był na algorytm Adam i dostrajanie jego współczynnika uczenia.

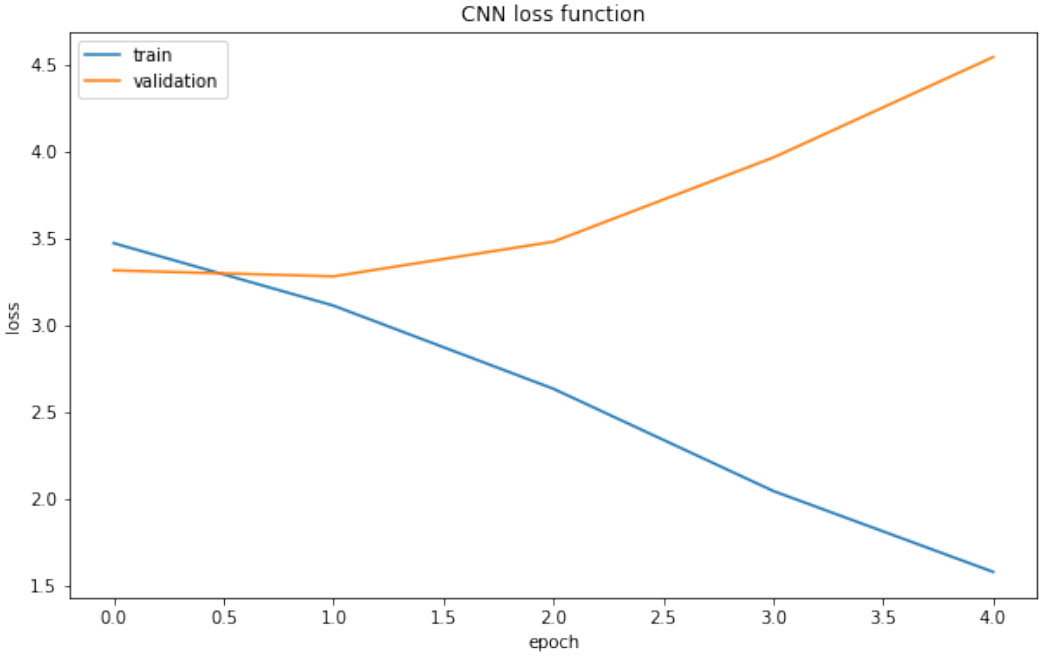
Dla najlepszego modelu przeprowadzone zostały dodatkowe badania. Poniżej zostały przedstawione wykresy funkcji straty oraz dokładności w trakcie uczenia dla zbiorów treningowego oraz walidacyjnego (7, 8). Zostały również wyznaczone miary precision, recall oraz f1-score dla poszczególnych klas. Skonstruowano także macierz pomyłek (9).

| num_units | dropout | optimizer | embedding | accuracy | f1 score macro |
|---|---|---|---|---|---|
| 64.000 | 0.50000 | sgd | 100.00 | 0.038728 | 0.0016572 |
| 64.000 | 0.25000 | adam | 100.00 | 0.13714 | 0.10494 |
| 128.00 | 0.25000 | adam | 100.00 | 0.14157 | 0.10851 |
| 64.000 | 0.25000 | sgd | 100.00 | 0.038715 | 0.0016570 |
| 128.00 | 0.50000 | sgd | 100.00 | 0.038728 | 0.0016571 |
| 64.000 | 0.50000 | adam | 100.00 | 0.12985 | 0.091153 |
| 128.00 | 0.25000 | sgd | 100.00 | 0.039072 | 0.0022198 |
| 128.00 | 0.50000 | adam | 100.00 | 0.13731 | 0.099725 |

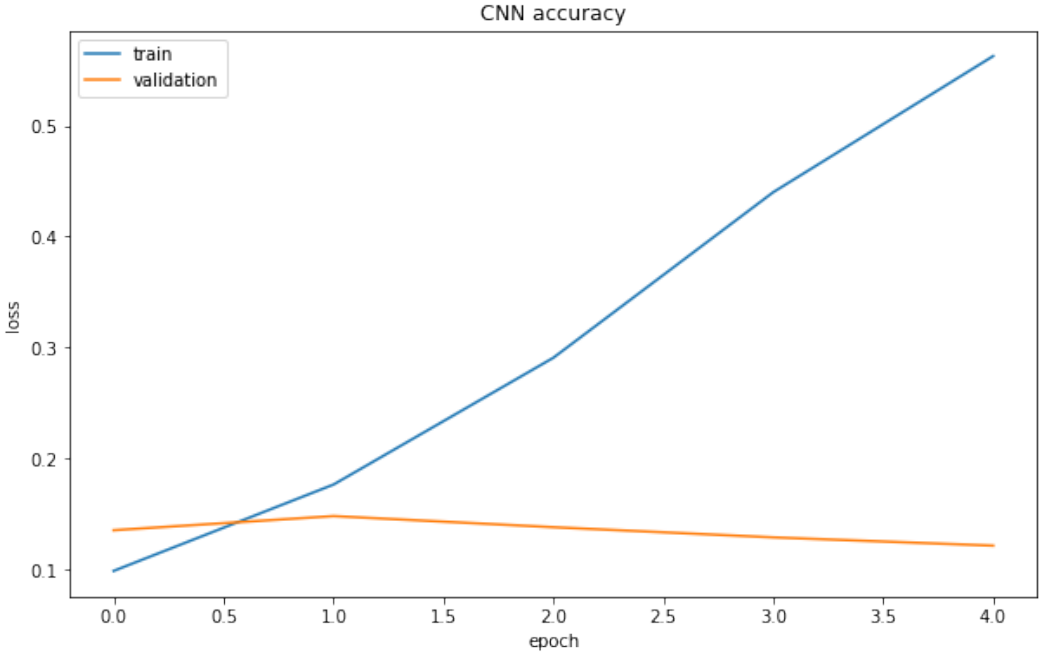
Rysunek 5: Wyniki eksperymentów dla sieci CNN z rozmiarem embeddingu 100.

| num_units | dropout | optimizer | embedding | accuracy | f1 score macro |
|---|---|---|---|---|---|
| 128.00 | 0.25000 | adam | 300.00 | 0.14800 | 0.12277 |
| 128.00 | 0.50000 | sgd | 300.00 | 0.040858 | 0.0030625 |
| 64.000 | 0.50000 | sgd | 300.00 | 0.040624 | 0.0031413 |
| 64.000 | 0.50000 | adam | 300.00 | 0.14082 | 0.10537 |
| 64.000 | 0.25000 | adam | 300.00 | 0.14579 | 0.11594 |
| 64.000 | 0.25000 | sgd | 300.00 | 0.040954 | 0.0030202 |
| 128.00 | 0.50000 | adam | 300.00 | 0.14361 | 0.11502 |
| 128.00 | 0.25000 | sgd | 300.00 | 0.040102 | 0.0029116 |

Rysunek 6: Wyniki eksperymentów dla sieci CNN z rozmiarem embeddingu 300.




Rysunek 7: Funkcja straty dla każdej epoki.

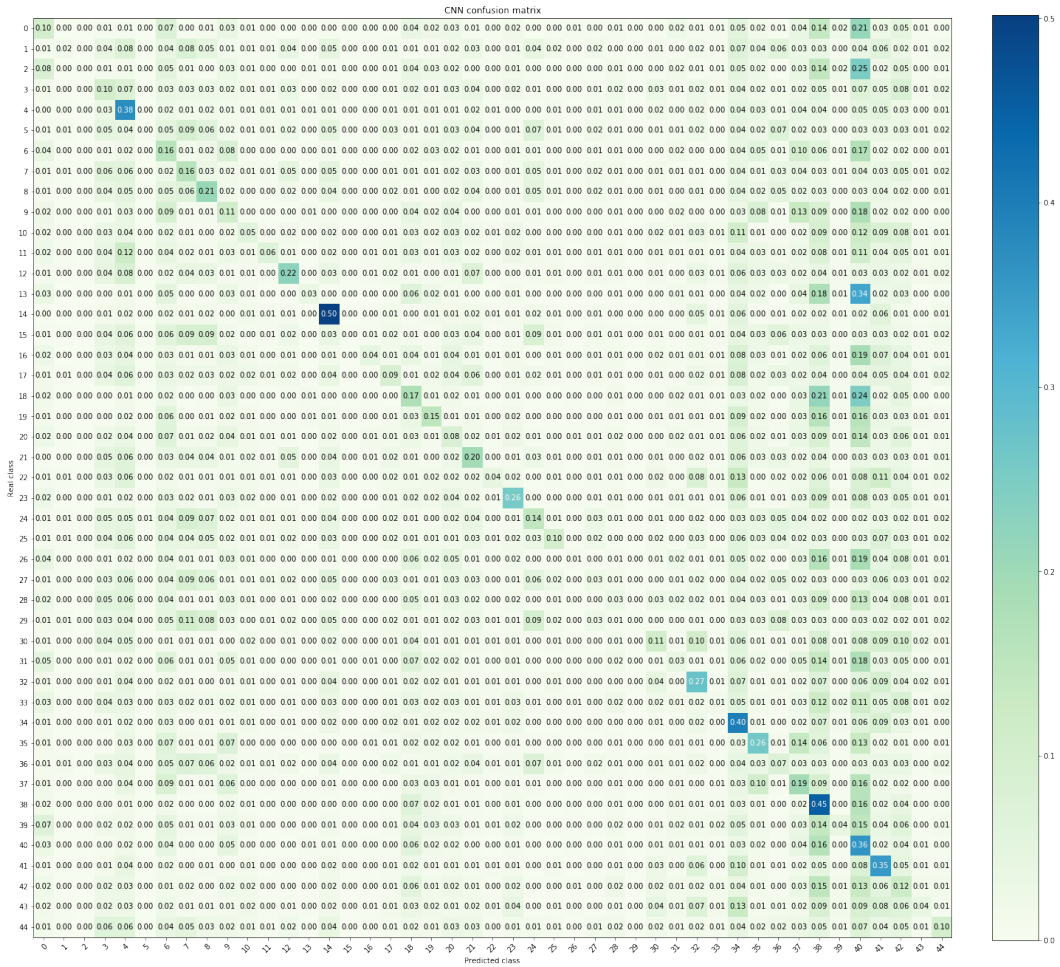



Rysunek 8: Dokładność dla każdej epoki.

Raport z metrykami dla CNN:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.12 | 0.10 | 0.11 | 1884 |
| 1 | 0.13 | 0.02 | 0.03 | 1486 |
| 2 | 0.14 | 0.00 | 0.00 | 1485 |
| 3 | 0.10 | 0.10 | 0.10 | 1728 |
| 4 | 0.18 | 0.38 | 0.25 | 1606 |
| 5 | 0.06 | 0.00 | 0.01 | 1347 |
| 6 | 0.10 | 0.16 | 0.12 | 1927 |
| 7 | 0.12 | 0.16 | 0.14 | 1384 |
| 8 | 0.19 | 0.21 | 0.20 | 1456 |
| 9 | 0.09 | 0.11 | 0.10 | 1799 |
| 10 | 0.11 | 0.05 | 0.07 | 1481 |
| 11 | 0.20 | 0.06 | 0.10 | 1282 |
| 12 | 0.24 | 0.22 | 0.23 | 1121 |
| 13 | 0.24 | 0.03 | 0.05 | 1611 |
| 14 | 0.27 | 0.50 | 0.35 | 1132 |
| 15 | 0.12 | 0.00 | 0.01 | 1074 |
| 16 | 0.17 | 0.04 | 0.07 | 1346 |
| 17 | 0.14 | 0.09 | 0.11 | 1387 |
| 18 | 0.14 | 0.17 | 0.15 | 1845 |
| 19 | 0.17 | 0.15 | 0.16 | 1514 |
| 20 | 0.08 | 0.08 | 0.08 | 1562 |
| 21 | 0.15 | 0.20 | 0.17 | 1120 |
| 22 | 0.17 | 0.04 | 0.07 | 1384 |
| 23 | 0.26 | 0.26 | 0.26 | 1386 |
| 24 | 0.15 | 0.14 | 0.15 | 1158 |
| 25 | 0.23 | 0.10 | 0.14 | 1232 |
| 26 | 0.07 | 0.01 | 0.01 | 1650 |
| 27 | 0.07 | 0.03 | 0.04 | 1305 |
| 28 | 0.06 | 0.03 | 0.04 | 1659 |
| 29 | 0.11 | 0.00 | 0.01 | 1025 |
| 30 | 0.17 | 0.11 | 0.13 | 1580 |
| 31 | 0.08 | 0.03 | 0.05 | 1800 |
| 32 | 0.23 | 0.27 | 0.25 | 1569 |
| 33 | 0.06 | 0.01 | 0.02 | 2032 |
| 34 | 0.19 | 0.40 | 0.26 | 2234 |
| 35 | 0.21 | 0.26 | 0.23 | 1661 |
| 36 | 0.10 | 0.07 | 0.08 | 1640 |
| 37 | 0.16 | 0.19 | 0.17 | 2324 |
| 38 | 0.18 | 0.45 | 0.25 | 2819 |
| 39 | 0.12 | 0.04 | 0.06 | 1794 |
| 40 | 0.10 | 0.36 | 0.16 | 2499 |
| 41 | 0.18 | 0.35 | 0.24 | 1870 |
| 42 | 0.08 | 0.12 | 0.10 | 2204 |
| 43 | 0.11 | 0.04 | 0.06 | 1555 |
| 44 | 0.19 | 0.10 | 0.13 | 1832 |
| accuracy | | | 0.15 | 72789 |
| macro avg | 0.15 | 0.14 | 0.12 | 72789 |
| weighted avg | 0.14 | 0.15 | 0.13 | 72789 |

Analizując otrzymane wyniki można zauważyć, że sieć przeucza się, a najlepsze wyniki na zbiorze walidacyjnym osiąga w stosunkowo wczesnych epokach. Dla niektórych klas wartości f1-score są bliskie zera co może świadczyć o tym, że model bardzo rzadko przypisuje daną klasę do czegokolwiek. Zauważyć to można również na macierzy pomyłek, gdzie dla niektórych klas predykowane są tak samo często wszystkie klasy. W zbiorze znajduje się kilka emoji, z którymi model radzi sobie dużo lepiej niż z pozostałymi.




Rysunek 9: Macierz pomyłek dla sieci CNN

# 6 RNN

Wykorzystana w zadaniu sieć RNN składa się z warstwy embeddingu, warstwy BiLSTM lub BiGRU, warstwy w pełni połączonej z funkcją aktywacji ReLU, warstway dropout oraz ostatniej warstwa w pełni połączonej z funkcją aktywacji softmax.

Eksperyment został przeprowadzony osobno dla sieci BiLSTM i BiGRU. W eksperymencie przebadane zostały przebadane takie parametry jak rozmiar embeddingu, liczba neuronów w warstwach rekurencyjnych, prawdopodobieństwo dropoutu oraz trzy wartości współczynnika uczenia się dla algorytmu optymalizacyjnego Adam. Dla każdej z prób rozmiar batcha wynosił 500, a liczba epok 10. Wyniki przedstawiono poniżej (tabele 10, 11, 12, 13).

Również tutaj decydujący wpływ okazał się mieć rozmiar embeddingu. Na ogół także lepsze okazywały się sieci z większym rozmiarem warstwy rekurencyjnej (128) oraz mniejszym dropoutem (0.25). Parametr uczenia algorymtu Adam nie miał znaczącego wpływu na wyniki. Z przeprowadzonych badań wynika, że model z warstwą GRU osiąga nieco lepszą skuteczność niż model z warstwą LSTM.

Dla najlepszych modeli przeprowadzone zostały dodatkowe badania. Poniżej zostały przedstawione wykresy funkcji straty oraz dokładności w trakcie uczenia dla zbiorów treningowego oraz walidacyjnego dla sieci BiLSTM (14, 15) oraz GRU (17, 18). Zostały również wyznaczone miary precision, recall oraz f1-score dla poszczególnych klas. Skonstruowano także macierze pomyłek (16, 19)

| dropout | embedding | layer | num_units | optimizer | accuracy | f1 |
|---|---|---|---|---|---|---|
| 0.50000 | 100.00 | lstm | 128.00 | 0.00010000 | 0.080933 | 0.032323 |
| 0.50000 | 100.00 | lstm | 64.000 | 0.00010000 | 0.084683 | 0.039036 |
| 0.25000 | 100.00 | lstm | 128.00 | 0.00010000 | 0.095564 | 0.062276 |
| 0.50000 | 100.00 | lstm | 128.00 | 0.010000 | 0.12114 | 0.080614 |
| 0.25000 | 100.00 | lstm | 128.00 | 0.010000 | 0.12279 | 0.091332 |
| 0.50000 | 100.00 | lstm | 128.00 | 0.0010000 | 0.12267 | 0.083177 |
| 0.25000 | 100.00 | lstm | 128.00 | 0.0010000 | 0.12468 | 0.087479 |
| 0.50000 | 100.00 | lstm | 64.000 | 0.010000 | 0.12610 | 0.093026 |
| 0.25000 | 100.00 | lstm | 64.000 | 0.00010000 | 0.087802 | 0.042270 |
| 0.50000 | 100.00 | lstm | 64.000 | 0.0010000 | 0.12411 | 0.088562 |
| 0.25000 | 100.00 | lstm | 64.000 | 0.010000 | 0.12321 | 0.083959 |
| 0.25000 | 100.00 | lstm | 64.000 | 0.0010000 | 0.11933 | 0.078092 |

Rysunek 10: Wyniki eksperymentów dla sieci LSTM z rozmiarem embeddingu 100.

| dropout | embedding | layer | num_units | optimizer | accuracy | f1 |
|---|---|---|---|---|---|---|
| 0.50000 | 300.00 | lstm | 64.000 | 0.00010000 | 0.097254 | 0.046890 |
| 0.25000 | 300.00 | lstm | 64.000 | 0.0010000 | 0.12834 | 0.094083 |
| 0.50000 | 300.00 | lstm | 128.00 | 0.00010000 | 0.093778 | 0.050158 |
| 0.50000 | 300.00 | lstm | 128.00 | 0.0010000 | 0.13093 | 0.092510 |
| 0.50000 | 300.00 | lstm | 64.000 | 0.0010000 | 0.12716 | 0.086542 |
| 0.25000 | 300.00 | lstm | 128.00 | 0.010000 | 0.12397 | 0.093948 |
| 0.25000 | 300.00 | lstm | 128.00 | 0.00010000 | 0.099122 | 0.055058 |
| 0.25000 | 300.00 | lstm | 64.000 | 0.010000 | 0.11948 | 0.081933 |
| 0.50000 | 300.00 | lstm | 128.00 | 0.010000 | 0.11651 | 0.075147 |
| 0.25000 | 300.00 | lstm | 128.00 | 0.0010000 | 0.13562 | 0.10438 |
| 0.50000 | 300.00 | lstm | 64.000 | 0.010000 | 0.11836 | 0.080927 |
| 0.25000 | 300.00 | lstm | 64.000 | 0.00010000 | 0.092871 | 0.044208 |

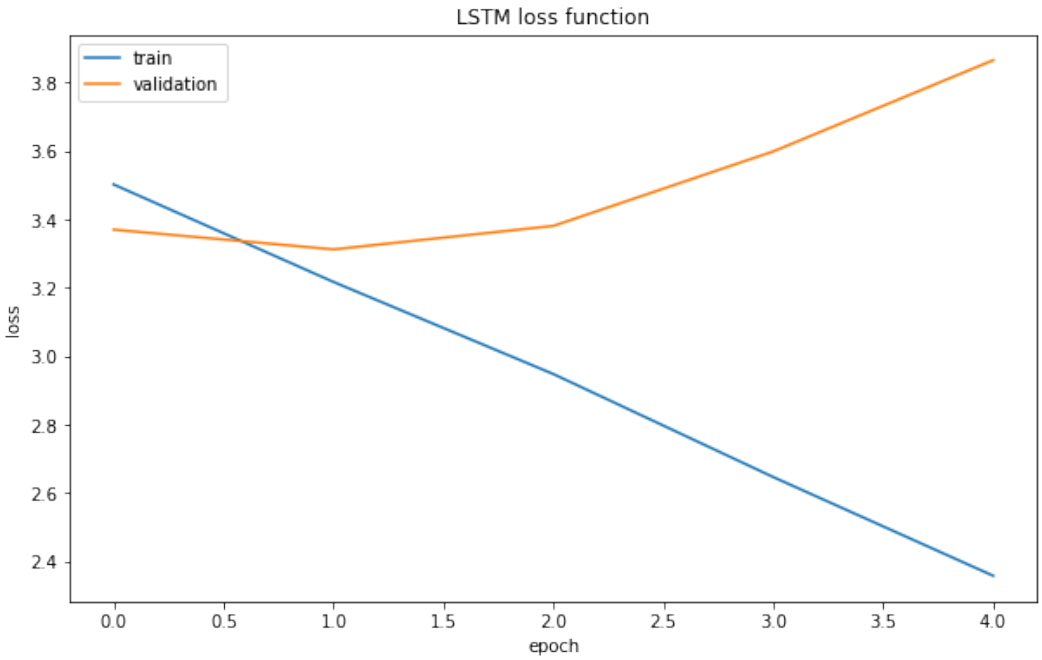
Rysunek 11: Wyniki eksperymentów dla sieci LSTM z rozmiarem embeddingu 300.

| dropout | embedding | layer | num_units | optimizer | accuracy | f1 |
|---|---|---|---|---|---|---|
| 0.25000 | 100.00 | gru | 128.00 | 0.010000 | 0.13124 | 0.097642 |
| 0.50000 | 100.00 | gru | 128.00 | 0.010000 | 0.12594 | 0.090482 |
| 0.25000 | 100.00 | gru | 64.000 | 0.00010000 | 0.086703 | 0.041289 |
| 0.25000 | 100.00 | gru | 64.000 | 0.010000 | 0.12918 | 0.094702 |
| 0.50000 | 100.00 | gru | 64.000 | 0.0010000 | 0.11525 | 0.073201 |
| 0.50000 | 100.00 | gru | 128.00 | 0.00010000 | 0.086455 | 0.041935 |
| 0.25000 | 100.00 | gru | 128.00 | 0.0010000 | 0.12797 | 0.095834 |
| 0.25000 | 100.00 | gru | 64.000 | 0.0010000 | 0.11617 | 0.075854 |
| 0.50000 | 100.00 | gru | 64.000 | 0.010000 | 0.12109 | 0.087015 |
| 0.50000 | 100.00 | gru | 64.000 | 0.00010000 | 0.085384 | 0.040392 |
| 0.50000 | 100.00 | gru | 128.00 | 0.0010000 | 0.12577 | 0.086641 |
| 0.25000 | 100.00 | gru | 128.00 | 0.00010000 | 0.088654 | 0.046059 |

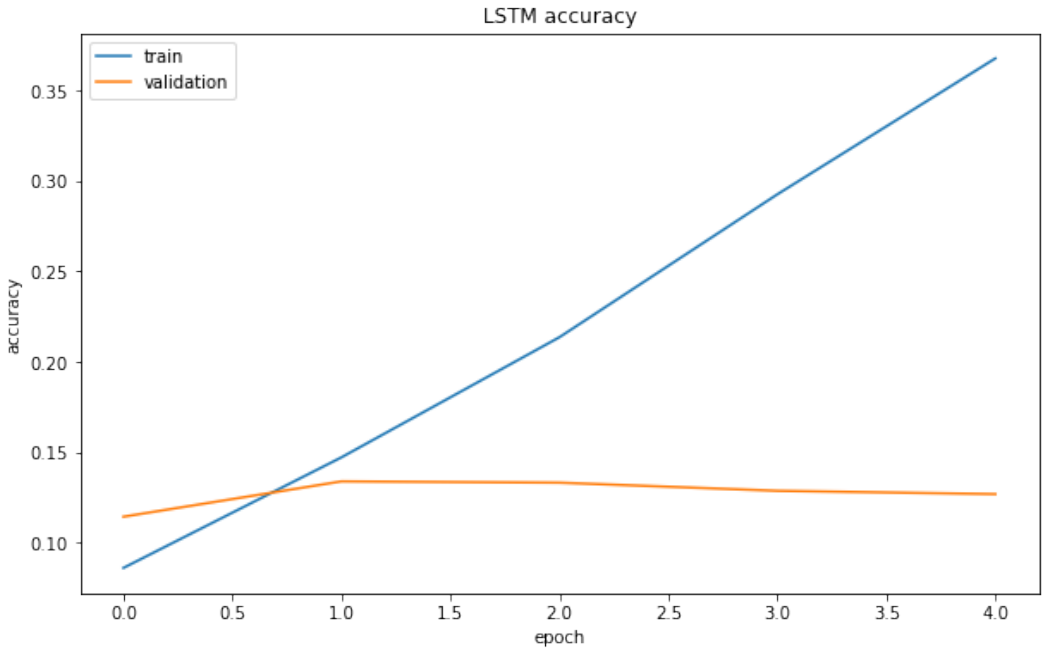
Rysunek 12: Wyniki eksperymentów dla sieci GRU z rozmiarem embeddingu 100.

| dropout | embedding | layer | num_units | optimizer | accuracy | f1 |
|---|---|---|---|---|---|---|
| 0.25000 | 300.00 | gru | 64.000 | 0.010000 | 0.12538 | 0.089826 |
| 0.25000 | 300.00 | gru | 128.00 | 0.00010000 | 0.092061 | 0.047032 |
| 0.25000 | 300.00 | gru | 64.000 | 0.0010000 | 0.13134 | 0.097849 |
| 0.50000 | 300.00 | gru | 128.00 | 0.0010000 | 0.13465 | 0.10142 |
| 0.50000 | 300.00 | gru | 128.00 | 0.010000 | 0.095372 | 0.062223 |
| 0.50000 | 300.00 | gru | 64.000 | 0.0010000 | 0.12501 | 0.084363 |
| 0.50000 | 300.00 | gru | 128.00 | 0.00010000 | 0.094355 | 0.050804 |
| 0.25000 | 300.00 | gru | 128.00 | 0.010000 | 0.10846 | 0.074246 |
| 0.50000 | 300.00 | gru | 64.000 | 0.00010000 | 0.091168 | 0.040206 |
| 0.25000 | 300.00 | gru | 128.00 | 0.0010000 | 0.13800 | 0.10880 |
| 0.25000 | 300.00 | gru | 64.000 | 0.00010000 | 0.091951 | 0.046160 |
| 0.50000 | 300.00 | gru | 64.000 | 0.010000 | 0.11573 | 0.080828 |

Rysunek 13: Wyniki eksperymentów dla sieci GRU z rozmiarem embeddingu 300.



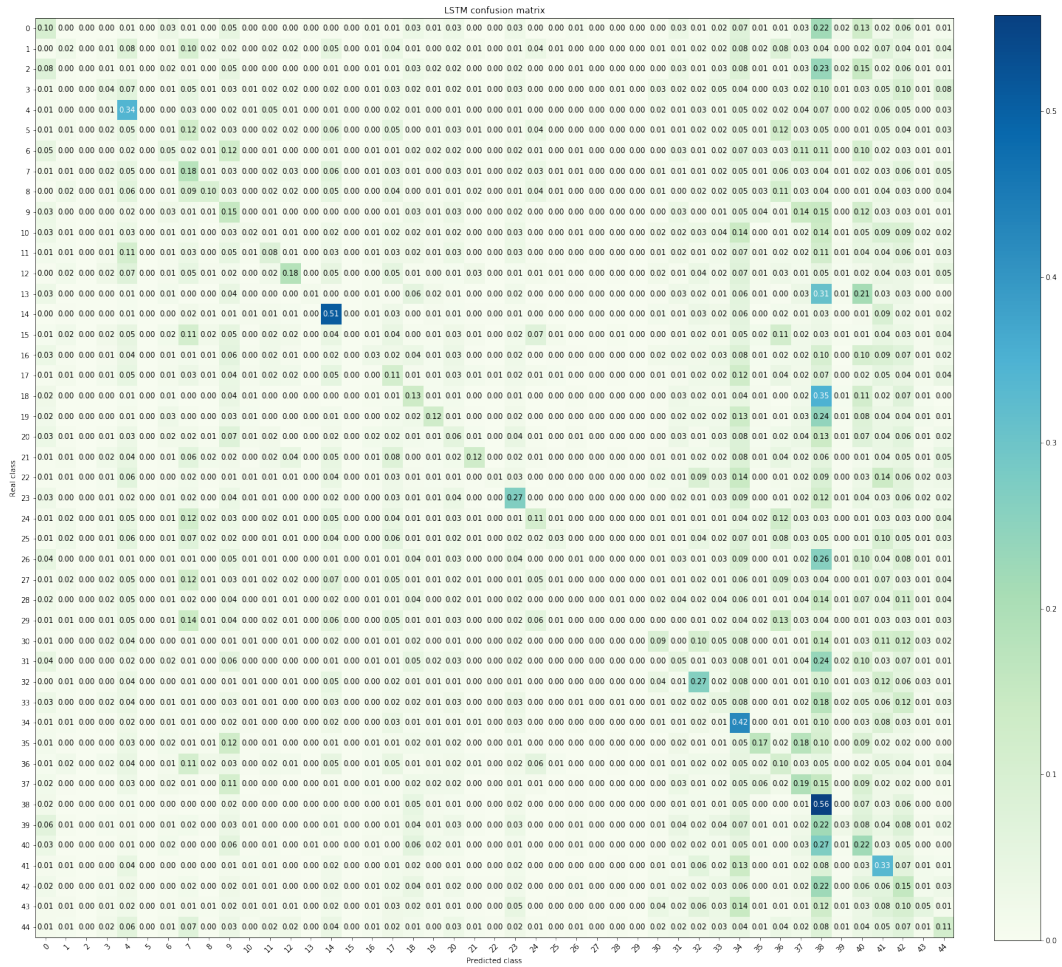

Rysunek 14: Funkcja straty dla każdej epoki.



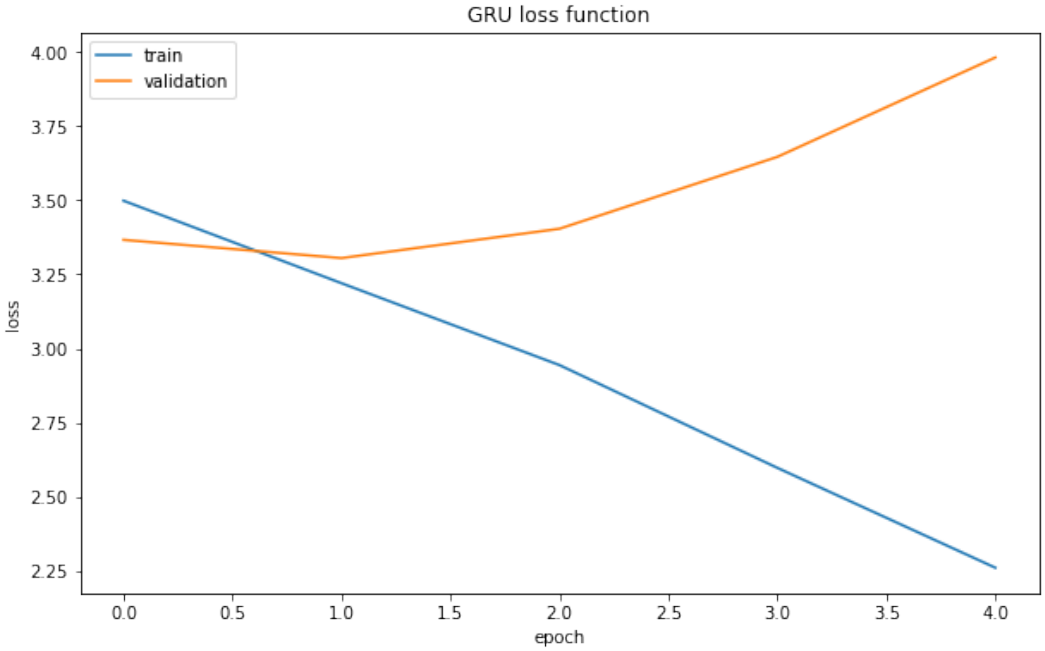

Rysunek 15: Dokładność dla każdej epoki.

Raport z metrykami dla BiLSTM:

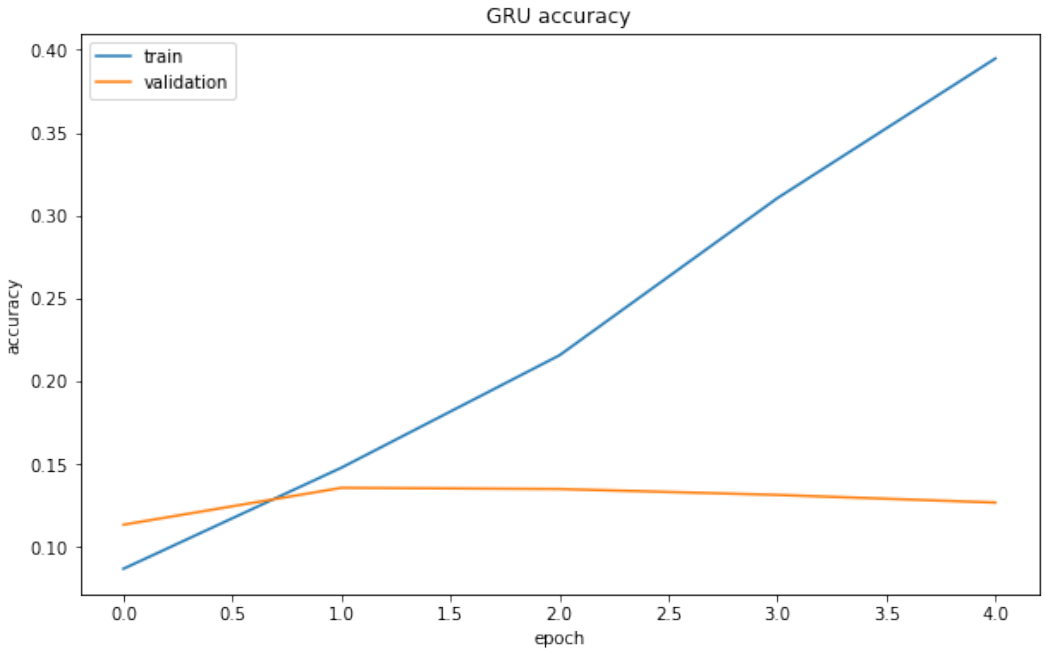
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.12 | 0.10 | 0.11 | 1884 |
| 1 | 0.07 | 0.02 | 0.03 | 1486 |
| 2 | 0.00 | 0.00 | 0.00 | 1485 |
| 3 | 0.09 | 0.04 | 0.05 | 1728 |
| 4 | 0.18 | 0.34 | 0.24 | 1606 |
| 5 | 0.00 | 0.00 | 0.00 | 1347 |
| 6 | 0.10 | 0.05 | 0.06 | 1927 |
| 7 | 0.10 | 0.18 | 0.13 | 1384 |
| 8 | 0.24 | 0.10 | 0.14 | 1456 |
| 9 | 0.09 | 0.15 | 0.11 | 1799 |
| 10 | 0.06 | 0.02 | 0.03 | 1481 |
| 11 | 0.11 | 0.08 | 0.09 | 1282 |
| 12 | 0.27 | 0.18 | 0.22 | 1121 |
| 13 | 0.18 | 0.01 | 0.02 | 1611 |
| 14 | 0.25 | 0.51 | 0.33 | 1132 |
| 15 | 0.00 | 0.00 | 0.00 | 1074 |
| 16 | 0.08 | 0.03 | 0.04 | 1346 |
| 17 | 0.09 | 0.11 | 0.10 | 1387 |
| 18 | 0.13 | 0.13 | 0.13 | 1845 |
| 19 | 0.21 | 0.12 | 0.16 | 1514 |
| 20 | 0.06 | 0.06 | 0.06 | 1562 |
| 21 | 0.28 | 0.12 | 0.17 | 1120 |
| 22 | 0.14 | 0.01 | 0.02 | 1384 |
| 23 | 0.20 | 0.27 | 0.23 | 1386 |
| 24 | 0.15 | 0.11 | 0.13 | 1158 |
| 25 | 0.18 | 0.03 | 0.05 | 1232 |
| 26 | 0.05 | 0.01 | 0.01 | 1650 |
| 27 | 0.00 | 0.00 | 0.00 | 1305 |
| 28 | 0.06 | 0.01 | 0.01 | 1659 |
| 29 | 0.00 | 0.00 | 0.00 | 1025 |
| 30 | 0.17 | 0.09 | 0.12 | 1580 |
| 31 | 0.06 | 0.05 | 0.06 | 1800 |
| 32 | 0.21 | 0.27 | 0.23 | 1569 |
| 33 | 0.06 | 0.05 | 0.05 | 2032 |
| 34 | 0.16 | 0.42 | 0.23 | 2234 |
| 35 | 0.25 | 0.17 | 0.20 | 1661 |
| 36 | 0.08 | 0.10 | 0.09 | 1640 |
| 37 | 0.16 | 0.19 | 0.17 | 2324 |
| 38 | 0.14 | 0.56 | 0.23 | 2819 |
| 39 | 0.10 | 0.03 | 0.04 | 1794 |
| 40 | 0.12 | 0.22 | 0.15 | 2499 |
| 41 | 0.15 | 0.33 | 0.21 | 1870 |
| 42 | 0.08 | 0.15 | 0.10 | 2204 |
| 43 | 0.11 | 0.05 | 0.07 | 1555 |
| 44 | 0.12 | 0.11 | 0.12 | 1832 |
| accuracy | | | 0.14 | 72789 |
| macro avg | 0.12 | 0.12 | 0.11 | 72789 |
| weighted avg | 0.12 | 0.14 | 0.11 | 72789 |




Rysunek 16: Macierz pomyłek dla sieci LSTM




Rysunek 17: Funkcja straty dla każdej epoki.

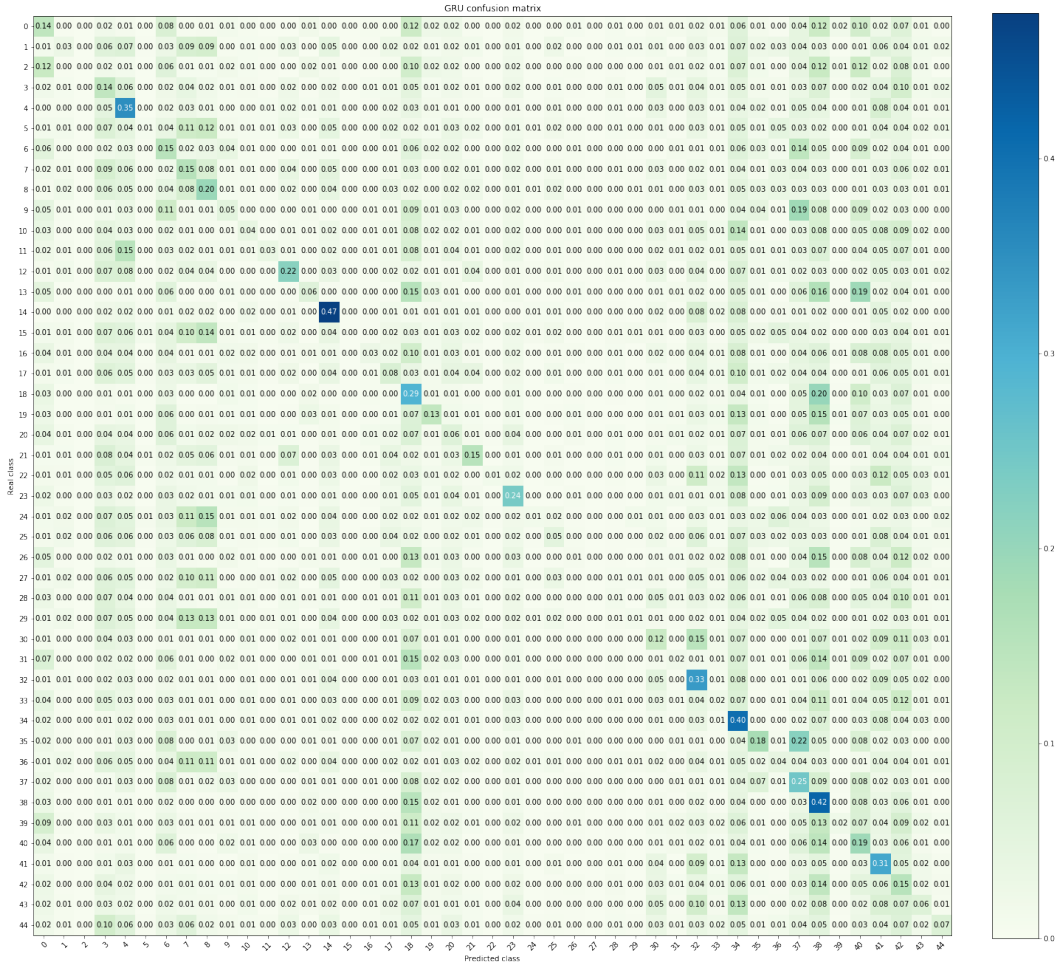



Rysunek 18: Dokładność dla każdej epoki.

Raport z metrykami dla BiGRU:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.12 | 0.14 | 0.13 | 1884 |
| 1 | 0.09 | 0.03 | 0.04 | 1486 |
| 2 | 0.33 | 0.00 | 0.00 | 1485 |
| 3 | 0.08 | 0.14 | 0.10 | 1728 |
| 4 | 0.19 | 0.35 | 0.24 | 1606 |
| 5 | 0.07 | 0.01 | 0.01 | 1347 |
| 6 | 0.10 | 0.15 | 0.12 | 1927 |
| 7 | 0.10 | 0.15 | 0.12 | 1384 |
| 8 | 0.12 | 0.20 | 0.15 | 1456 |
| 9 | 0.11 | 0.05 | 0.07 | 1799 |
| 10 | 0.09 | 0.04 | 0.05 | 1481 |
| 11 | 0.19 | 0.03 | 0.05 | 1282 |
| 12 | 0.23 | 0.22 | 0.22 | 1121 |
| 13 | 0.15 | 0.06 | 0.09 | 1611 |
| 14 | 0.28 | 0.47 | 0.35 | 1132 |
| 15 | 0.00 | 0.00 | 0.00 | 1074 |
| 16 | 0.11 | 0.03 | 0.05 | 1346 |
| 17 | 0.10 | 0.08 | 0.09 | 1387 |
| 18 | 0.10 | 0.29 | 0.15 | 1845 |
| 19 | 0.18 | 0.13 | 0.15 | 1514 |
| 20 | 0.06 | 0.06 | 0.06 | 1562 |
| 21 | 0.22 | 0.15 | 0.18 | 1120 |
| 22 | 0.29 | 0.01 | 0.03 | 1384 |
| 23 | 0.21 | 0.24 | 0.22 | 1386 |
| 24 | 0.15 | 0.01 | 0.03 | 1158 |
| 25 | 0.12 | 0.05 | 0.07 | 1232 |
| 26 | 0.06 | 0.01 | 0.01 | 1650 |
| 27 | 0.00 | 0.00 | 0.00 | 1305 |
| 28 | 0.06 | 0.01 | 0.02 | 1659 |
| 29 | 0.11 | 0.01 | 0.02 | 1025 |
| 30 | 0.13 | 0.12 | 0.13 | 1580 |
| 31 | 0.07 | 0.02 | 0.03 | 1800 |
| 32 | 0.17 | 0.33 | 0.23 | 1569 |
| 33 | 0.06 | 0.02 | 0.03 | 2032 |
| 34 | 0.16 | 0.40 | 0.23 | 2234 |
| 35 | 0.23 | 0.18 | 0.20 | 1661 |
| 36 | 0.08 | 0.04 | 0.05 | 1640 |
| 37 | 0.14 | 0.25 | 0.18 | 2324 |
| 38 | 0.18 | 0.42 | 0.25 | 2819 |
| 39 | 0.12 | 0.02 | 0.03 | 1794 |
| 40 | 0.12 | 0.19 | 0.15 | 2499 |
| 41 | 0.16 | 0.31 | 0.21 | 1870 |
| 42 | 0.08 | 0.15 | 0.10 | 2204 |
| 43 | 0.10 | 0.06 | 0.08 | 1555 |
| 44 | 0.24 | 0.07 | 0.11 | 1832 |
| accuracy | | | 0.14 | 72789 |
| macro avg | 0.14 | 0.13 | 0.11 | 72789 |
| weighted avg | 0.13 | 0.14 | 0.11 | 72789 |

Również tutaj tak jak w przypadku sieci CNN analizując otrzymane wyniki można zauważyć, że sieć przeucza się, a najlepsze wyniki na zbiorze walidacyjnym osiąga w stosunkowo wczesnych epokach. Dla niektórych klas wartości f1-score są bliskie zera co może świadczyć o tym, że model bardzo rzadko przypisuje daną klasę do czegokolwiek. Zauważyć to można również na macierzy pomyłek, gdzie dla niektórych klas predykowane są tak samo często wszystkie klasy. W zbiorze znajduje się kilka emoji, z którymi model radzi sobie dużo lepiej



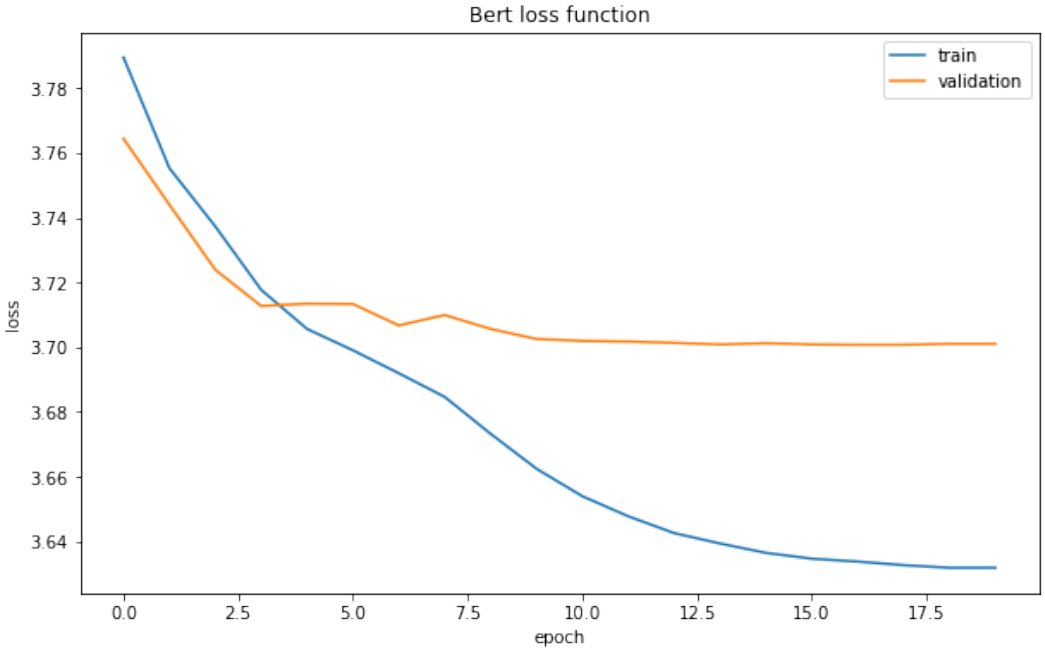

Rysunek 19: Macierz pomyłek dla sieci GRU

niż z pozostałymi. Z otrzymanych wyników można wnioskować, że model BiLSTM radzi sobie minimalnie gorzej niż BiGRU.

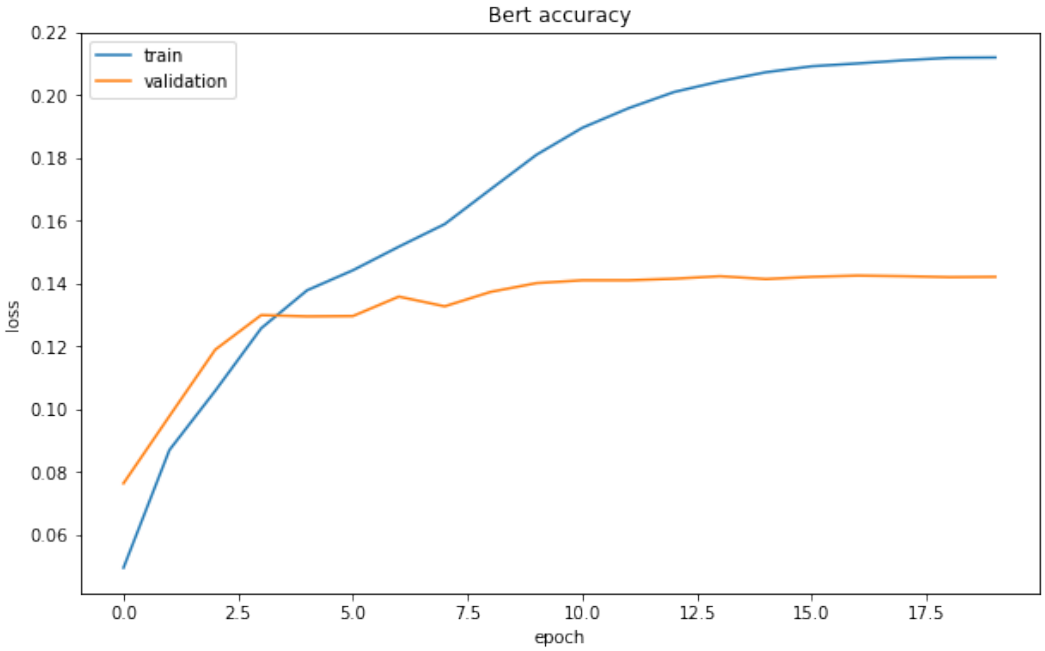
# 7 BERT

W zadaniu został wykorzystany przetrenowany model BERT, który został połączony z warstwami w pełni połączonymi – pierwszą z 768 neuronami i funkcją aktywacji tangensem hiperbolicznym, drugą z wyjściową liczbą klas oraz aktywacją softmax. Warstwy te dodatkowo zostały przedzielone warstwą dropout o współczynniku 0.5. Ze względu na czas potrzebny do uczenia modelu dokonano tylko dwóch prób, pierwszej z optymalizatorem Adam i domyślnym parametrem uczenia się oraz drugi z dynamicznym parametrem uczenia się. Pierwszy z eksperymentów zakończył się porażką – model praktycznie się nie uczył, w drugim natomiast ze względu na zmniejszony współczynnik uczenia model zbiegał bardzo wolno. Ostatecznie uzyskano dokładność na poziomie **0.14** oraz f1-score macro **0.07**.

Poniżej zostały przedstawione wykresy funkcji straty oraz dokładności w trakcie uczenia dla zbiorów treningowego oraz walidacyjnego (20, 21). Zostały również wyznaczone miary precision, recall oraz f1-score dla poszczególnych klas. Skonstruowano także macierz pomyłek (19).




Rysunek 20: Funkcja straty dla każdej epoki.

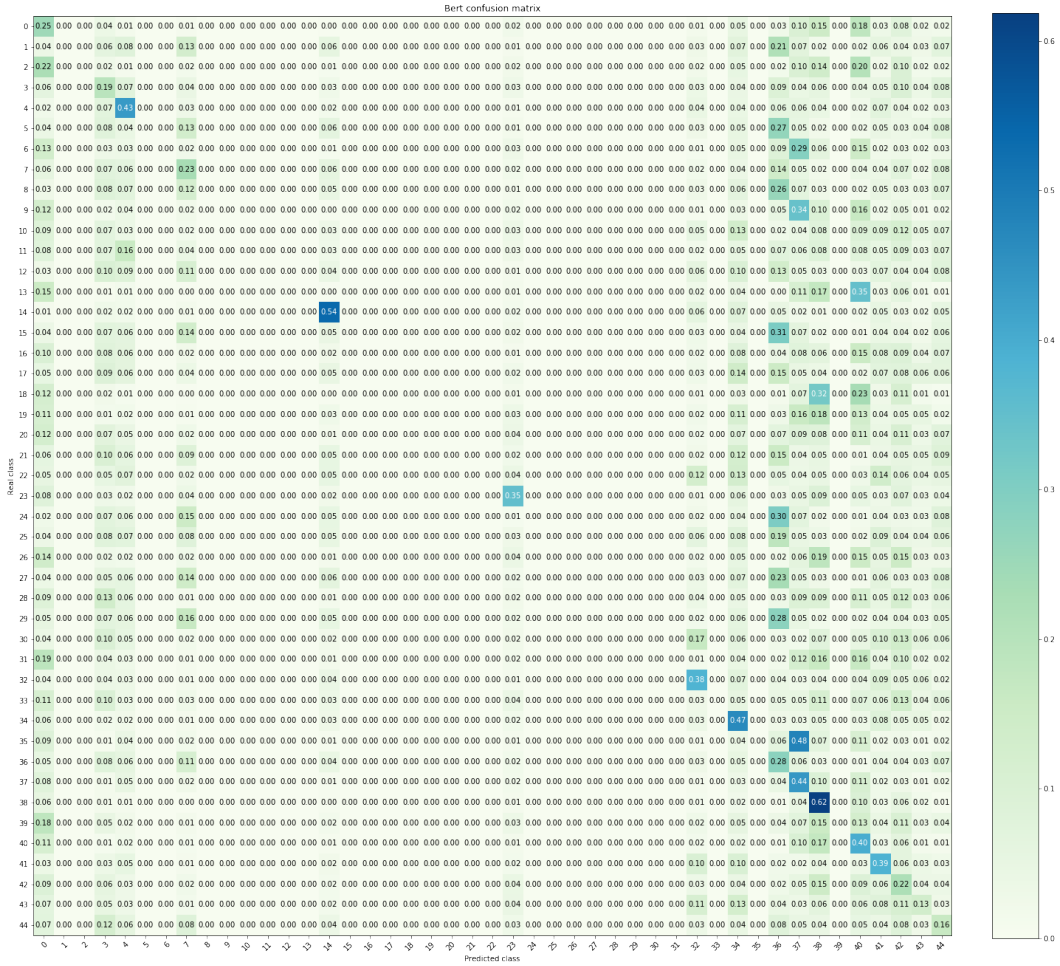



Rysunek 21: Dokładność dla każdej epoki.

Raport z metrykami dla BERT:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.07 | 0.25 | 0.11 | 1884 |
| 1 | 0.00 | 0.00 | 0.00 | 1486 |
| 2 | 0.00 | 0.00 | 0.00 | 1485 |
| 3 | 0.08 | 0.19 | 0.12 | 1728 |
| 4 | 0.19 | 0.43 | 0.26 | 1606 |
| 5 | 0.00 | 0.00 | 0.00 | 1347 |
| 6 | 0.00 | 0.00 | 0.00 | 1927 |
| 7 | 0.10 | 0.23 | 0.14 | 1384 |
| 8 | 0.00 | 0.00 | 0.00 | 1456 |
| 9 | 0.00 | 0.00 | 0.00 | 1799 |
| 10 | 0.00 | 0.00 | 0.00 | 1481 |
| 11 | 0.00 | 0.00 | 0.00 | 1282 |
| 12 | 0.00 | 0.00 | 0.00 | 1121 |
| 13 | 0.00 | 0.00 | 0.00 | 1611 |
| 14 | 0.25 | 0.54 | 0.34 | 1132 |
| 15 | 0.00 | 0.00 | 0.00 | 1074 |
| 16 | 0.00 | 0.00 | 0.00 | 1346 |
| 17 | 0.00 | 0.00 | 0.00 | 1387 |
| 18 | 0.00 | 0.00 | 0.00 | 1845 |
| 19 | 0.00 | 0.00 | 0.00 | 1514 |
| 20 | 0.00 | 0.00 | 0.00 | 1562 |
| 21 | 0.00 | 0.00 | 0.00 | 1120 |
| 22 | 0.00 | 0.00 | 0.00 | 1384 |
| 23 | 0.24 | 0.35 | 0.28 | 1386 |
| 24 | 0.00 | 0.00 | 0.00 | 1158 |
| 25 | 0.00 | 0.00 | 0.00 | 1232 |
| 26 | 0.00 | 0.00 | 0.00 | 1650 |
| 27 | 0.00 | 0.00 | 0.00 | 1305 |
| 28 | 0.00 | 0.00 | 0.00 | 1659 |

| | | | | |
|---|---|---|---|---|
| 29 | 0.00 | 0.00 | 0.00 | 1025 |
| 30 | 0.00 | 0.00 | 0.00 | 1580 |
| 31 | 0.00 | 0.00 | 0.00 | 1800 |
| 32 | 0.20 | 0.38 | 0.27 | 1569 |
| 33 | 0.00 | 0.00 | 0.00 | 2032 |
| 34 | 0.20 | 0.47 | 0.28 | 2234 |
| 35 | 0.00 | 0.00 | 0.00 | 1661 |
| 36 | 0.08 | 0.28 | 0.12 | 1640 |
| 37 | 0.15 | 0.44 | 0.22 | 2324 |
| 38 | 0.22 | 0.62 | 0.33 | 2819 |
| 39 | 0.00 | 0.00 | 0.00 | 1794 |
| 40 | 0.14 | 0.40 | 0.21 | 2499 |
| 41 | 0.17 | 0.39 | 0.24 | 1870 |
| 42 | 0.09 | 0.22 | 0.13 | 2204 |
| 43 | 0.09 | 0.13 | 0.10 | 1555 |
| 44 | 0.09 | 0.16 | 0.12 | 1832 |
| accuracy | | | 0.14 | 72789 |
| macro avg | 0.05 | 0.12 | 0.07 | 72789 |
| weighted avg | 0.06 | 0.14 | 0.08 | 72789 |




Rysunek 22: Macierz pomyłek dla sieci BERT

W porównaniu z poprzednimi sieciami model BERT nie przeucza się tak bardzo, jednak końcowe wyniki na zbiorze testowym są wręcz gorsze niż dla pozostałych modeli. W

przypadku tej sieci również dla niektórych klas wartości f1-score są bliskie zera, częściej niż dla pozostałych modeli. Model często przypisuje klasę wybraną z dość małej puli klas, niektórych klas w ogóle nie przypisuje żadnym przypadkom. Wskazują na to wyniki widoczne na macierzy pomyłek jak również niska wartośc metryki recall. Tutaj znowu wyróżnia się kilka klas, które są klasyfikowane dużo lepiej od pozostałych.

# 8 Porównanie wyników

Poniżej przedstawiono porównanie dokładności oraz f1-score macro dla najlepszych sieci z każdej architektury.

| | accuracy | f1-score |
|---|---|---|
| cnn | 0.15 | 0.12 |
| bilstm | 0.14 | 0.11 |
| bigru | 0.14 | 0.11 |
| bert | 0.14 | 0.07 |

Najlepsze rezultaty, zarówno pod względem dokładności jak i f1-score, zostały osiągnięte za pomocą sieci CNN. Najgorzej poradził sobie model BERT. Ogółem sieci popełniały podobne błędy. Duża liczba klas była rzadko albo nawet wcale przypisywana przykładom. Kilka klas było przypisywanych bardzo często. Osiągnięte wyniki, nie są świetne, ale również nie bardzo złe. Uwagę trzeba zwrócić na dużą liczbę klas, na to że część emoji jest do siebie bardzo podobna, jak również na to, że czasami do kontekstu pasuje kilka emoji.

# 9 Predykcja grup

W celu poprawienia jakości predykcji została dokonana klasyfikacja sześciu grup podobnych emoji. Stoi za tym intuicja, że czasami emoji w kontekście mogą być wymienne między sobą. Predykcja grupy podobnych emoji może pozwolić zatem na znalezienie kilku emoji, które pasują do tekstu w podobny sposób. Poniżej zostały przedstawione wyniki sieci dla opisanych wcześniej grup. Zastosowano tutaj architektury znalezione w poprzednim problemie.

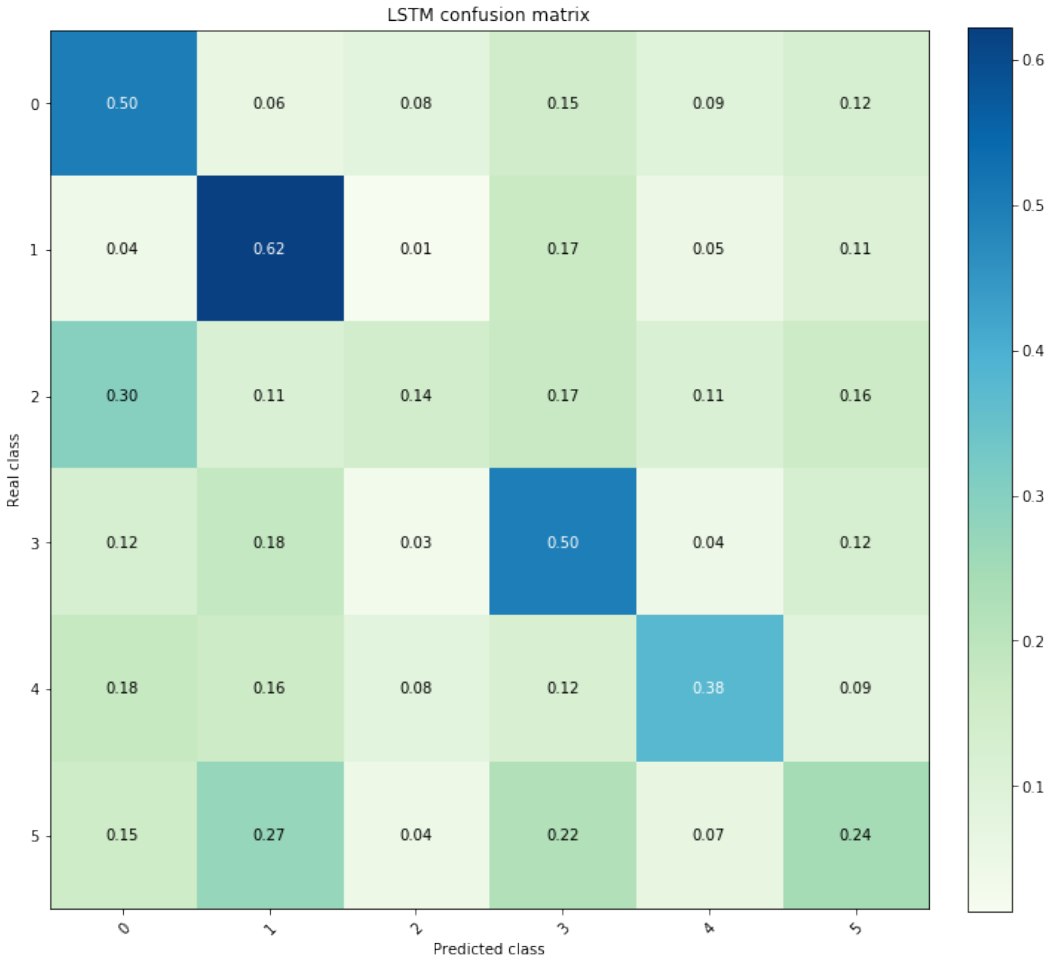
## 9.1 CNN

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.34 | 0.61 | 0.44 | 10424 |
| 1 | 0.58 | 0.53 | 0.55 | 16483 |
| 2 | 0.33 | 0.17 | 0.22 | 8477 |
| 3 | 0.46 | 0.45 | 0.45 | 14625 |
| 4 | 0.44 | 0.34 | 0.39 | 7711 |
| 5 | 0.35 | 0.33 | 0.34 | 15069 |
| accuracy | | | 0.42 | 72789 |
| macro avg | 0.41 | 0.40 | 0.40 | 72789 |
| weighted avg | 0.43 | 0.42 | 0.42 | 72789 |


Rysunek 23: Macierz pomyłek dla CNN dla grup.

## 9.2 BiLSTM

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.38 | 0.50 | 0.43 | 10424 |
| 1 | 0.52 | 0.62 | 0.57 | 16483 |
| 2 | 0.30 | 0.14 | 0.19 | 8477 |
| 3 | 0.42 | 0.50 | 0.46 | 14625 |
| 4 | 0.40 | 0.38 | 0.39 | 7711 |
| 5 | 0.35 | 0.24 | 0.29 | 15069 |
| accuracy | | | 0.42 | 72789 |
| macro avg | 0.39 | 0.40 | 0.39 | 72789 |
| weighted avg | 0.41 | 0.42 | 0.40 | 72789 |

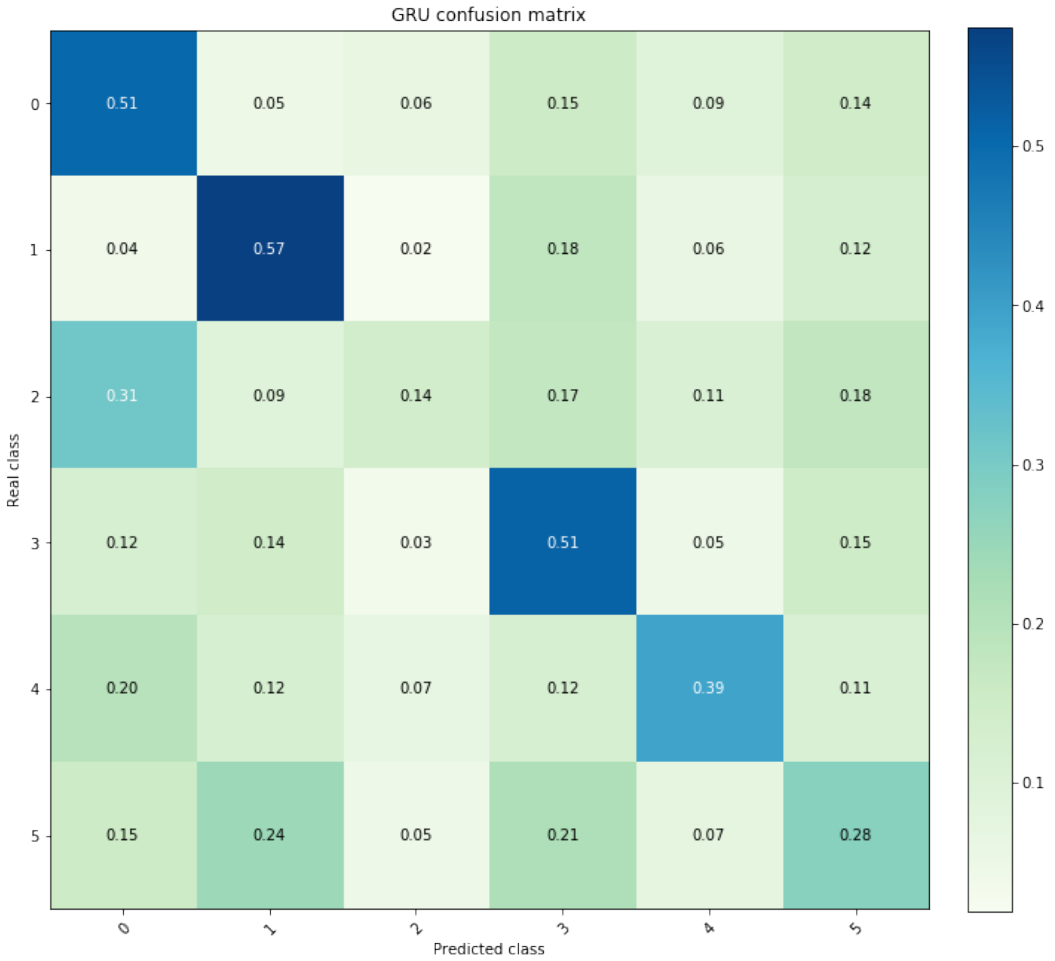



Rysunek 24: Macierz pomyłek dla BiLSTM dla grup.

### 9.3 BiGRU

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.37 | 0.51 | 0.43 | 10424 |
| 1 | 0.54 | 0.57 | 0.56 | 16483 |
| 2 | 0.31 | 0.14 | 0.19 | 8477 |
| 3 | 0.42 | 0.51 | 0.46 | 14625 |
| 4 | 0.40 | 0.39 | 0.40 | 7711 |
| 5 | 0.34 | 0.28 | 0.31 | 15069 |
| accuracy | | | 0.42 | 72789 |
| macro avg | 0.40 | 0.40 | 0.39 | 72789 |
| weighted avg | 0.41 | 0.42 | 0.41 | 72789 |

Wszystkie sieci osiągnęły dokładność na poziomie 0.42. Najlepiej po raz kolejny poradziła sobie sieć CNN uzyskują f1-score równy 0.42 (BiLSTM - 0.40, BiGRU - 0.41). Dla trzech klas modele osiągnęły dość dobre wyniki, natomiast jedna klasa była klasyfikowana częściej jako inna (nr 2 jako nr 0) niż jako ona sama. Skuteczność poprawiła się jednak znacząco w stosunku do modeli dla pojedynczych emoji.




Rysunek 25: Macierz pomyłek dla BiGRU dla grup.

## 10 Miękka ewaluacja

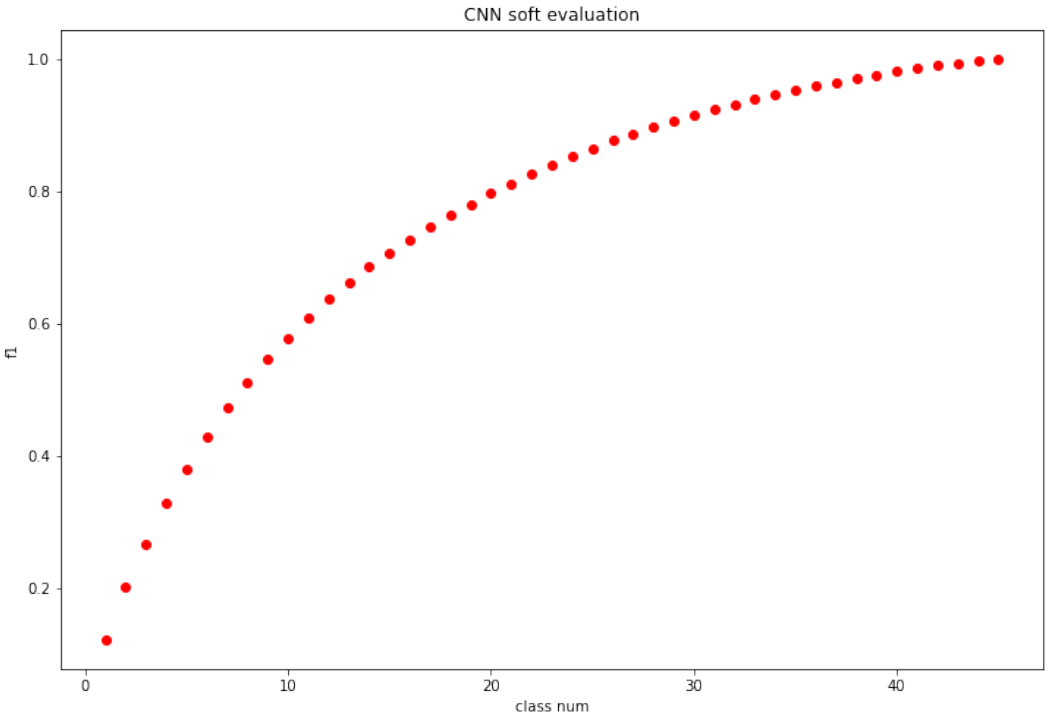
W celu sprawdzenia tego jak bardzo modele predykujące pojedyncze emoji się mylą zastosowano tzw. miękką ewaluację. Polegała ona na tym, że za wynik poprawny klasyfikatora uznawano kiedy przewidział on poprawną klasę na co najwyżej $n$-tym miejscu. Na wykresach poniżej (26, 27, 28, 29) przedstawiono dla poszczególnych klasyfikatorów wyniki f1-score w zależności od $n$. Można zauważyć, że dla początkowych $n$ wartości f1-score rosną szybciej, co odpowiada intuicji, że model przewiduje poprawną klasę częściej na miejscach bliższych. W praktyce zatem można proponować większą liczbę emoji, żeby osiągnąć daną skuteczność. Poniższa tabelka przedstawia wartości dokładności oraz f1-score dla wybranych $n$.

Dokładność:

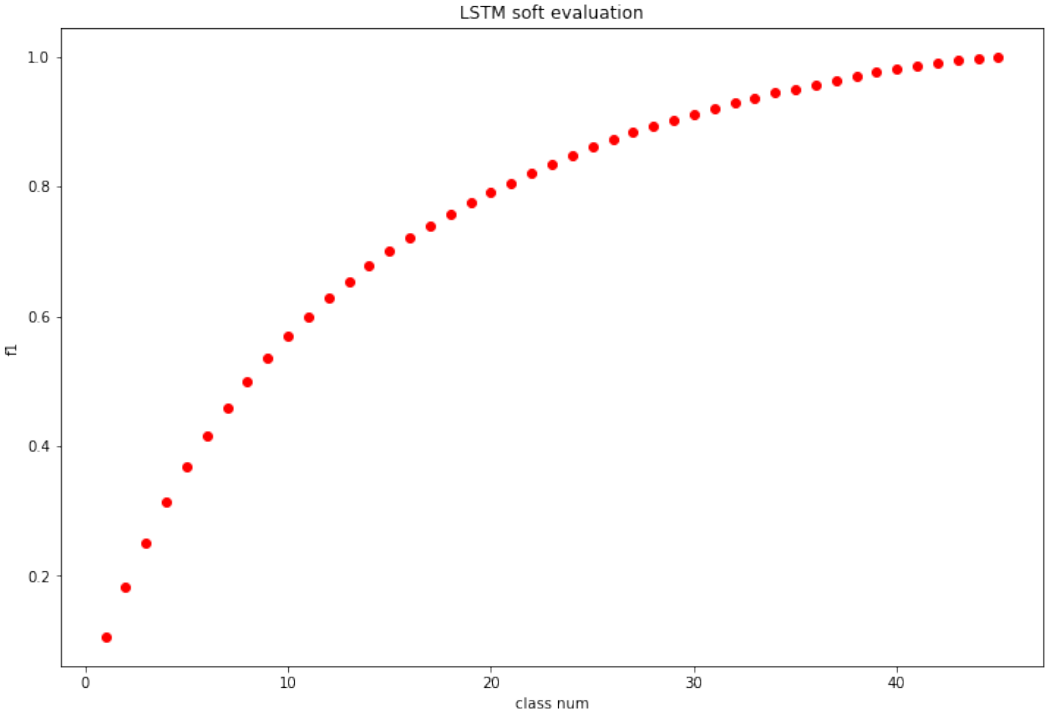
| n | 5 | 10 | 15 |
|---|---|---|---|
| cnn | 0.44 | 0.61 | 0.73 |
| bilstm | 0.42 | 0.60 | 0.72 |
| bigru | 0.42 | 0.59 | 0.72 |
| bert | 0.38 | 0.53 | 0.65 |

F1-score:

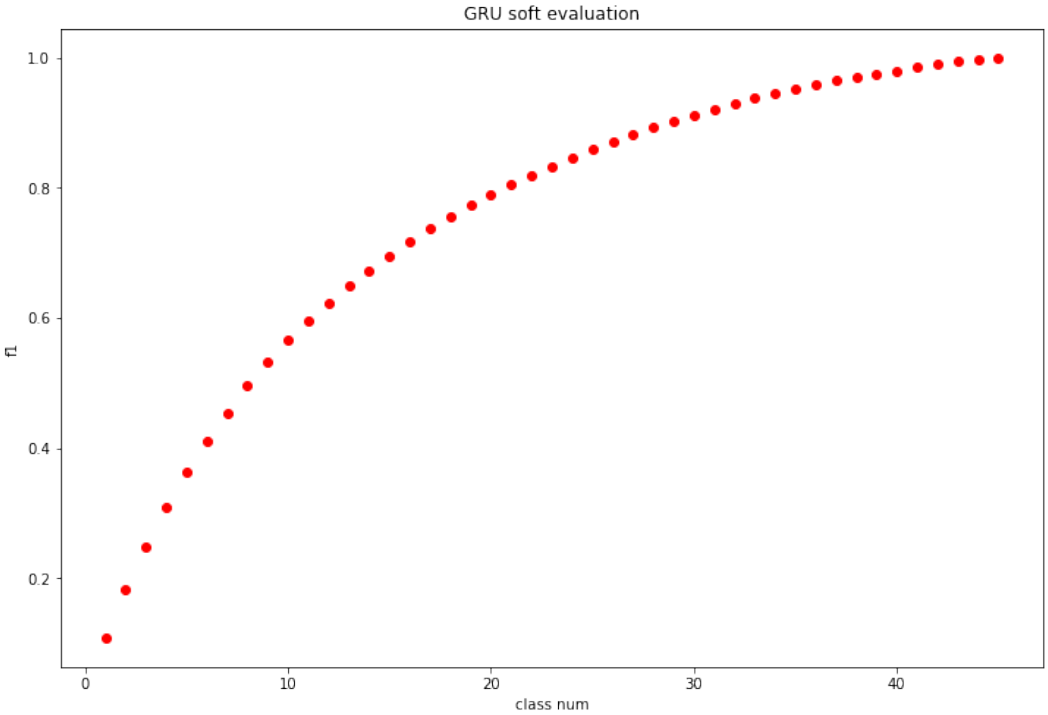
| n | 5 | 10 | 15 |
|---|---|---|---|
| cnn | 0.43 | 0.61 | 0.73 |
| bilstm | 0.42 | 0.60 | 0.72 |
| bigru | 0.41 | 0.59 | 0.72 |
| bert | 0.32 | 0.52 | 0.66 |



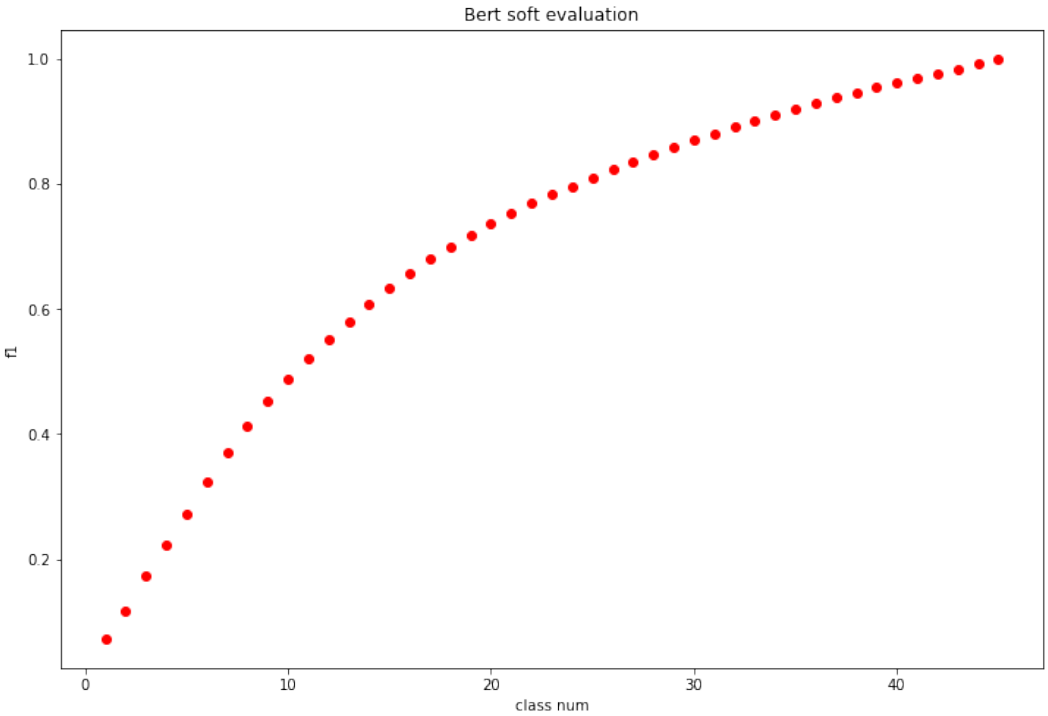

Rysunek 26: Miękka ewaluacja dla CNN.




Rysunek 27: Miękka ewaluacja dla BiLSTM.




Rysunek 28: Miękka ewaluacja dla GRU.




Rysunek 29: Miękka ewaluacja dla BERT.

## 11 Podsumowanie

Zadanie predykcji emoji jest trudne ze względu na to że często pojawiają się one w różnych znaczeniach, ludzie stosują je odmiennie. Niektóre z nich pozostają jednak bardzo charakterystyczne i modele były w stanie bardzo dobrze poradzić sobie z ich predykcją. Inne z kolei pojawiają się w szerszym kontekście i klasyfikowane są dużo słabiej. Część emoji była bardzo rzadko wybierana jako klasa dla przykładów. W poprawie klasyfikacji może pomóc zastosowanie klasteryzacji i przewidywania grup zamiast pojedynczych emoji albo proponowanie kilku najbardziej prawdopodobnych dla zdania emoji. Najlepszym z przebadanych modeli w zasadzie w każdej metodzie okazał się model oparty o trzy równoległe sieci konwolucyjne. Zachęca to do stosowania tego typu modeli w klasyfikacji tekstów, gdyż mogą sobie radzić, jak to pokazały badania, lepiej od modeli rekurencyjnych.

## Literatura


[1] F. Barbieri, J. Camacho-Collados, F. Ronzano, L. Espinosa-Anke, M. Ballesteros, V. Basile, V. Patti, and H. Saggion, "SemEval 2018 task 2: Multilingual emoji prediction," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, (New Orleans, Louisiana), pp. 24–33, Association for Computational Linguistics, June 2018.

[2] F. Barbieri, M. Ballesteros, and H. Saggion, "Are emojis predictable?," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 105–111, Association for Computational Linguistics, Apr. 2017.

[3] Ç. Çöltekin and T. Rama, "Tübingen-oslo at SemEval-2018 task 2: SVMs perform better than RNNs in emoji prediction," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, (New Orleans, Louisiana), pp. 34–38, Association for Computational Linguistics, June 2018.

[4] C. Baziotis, A. Nikolaos, A. Kolovou, G. Paraskevopoulos, N. Ellinas, and A. Potamianos, "NTUA-SLP at SemEval-2018 task 2: Predicting emojis using RNNs with context-aware attention," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, (New Orleans, Louisiana), pp. 438–444, Association for Computational Linguistics, June 2018.

[5] M. Liu, "EmoNLP at SemEval-2018 task 2: English emoji prediction with gradient boosting regression tree method and bidirectional LSTM," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, (New Orleans, Louisiana), pp. 390–394, Association for Computational Linguistics, June 2018.

[6] A. Jacovi, O. Sar Shalom, and Y. Goldberg, "Understanding convolutional neural networks for text classification," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, (Brussels, Belgium), pp. 56–65, Association for Computational Linguistics, Nov. 2018.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June 2019.

[9] S. Chi, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," 05 2019.