# **Model 1**

Wiele warstw różnych typów, dane mało przetworzone

model = Sequential()

model.add(Embedding(vocab\_size+1, embedding\_dim, input\_length=max\_lenght))

model.add(GRU(128, return\_sequences=True))

model.add(Dropout(rate=0.2))

model.add(GRU(64, return\_sequences=True))

model.add(Dropout(rate=0.2))

model.add(GRU(32, return\_sequences=True))

model.add(Conv1D(100, 6))

model.add(MaxPooling1D(6))

model.add(Dropout(rate=0.2))

model.add(GRU(16))

model.add(Dropout(rate=0.2))

model.add(Flatten())

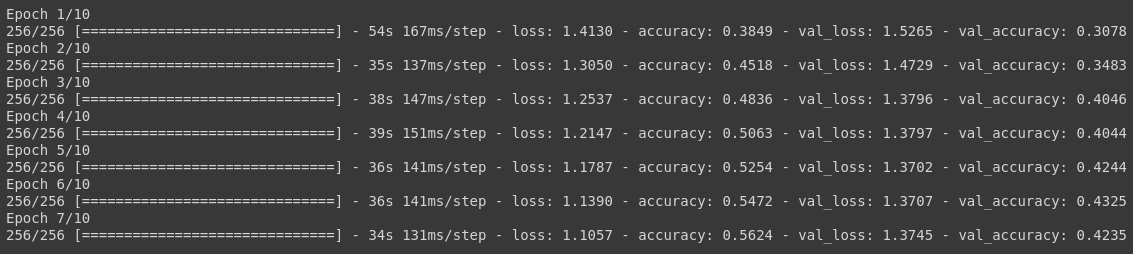
model.add(Dense(125, activation="relu"))

model.add(Dense(25, activation="relu"))

model.add(Dense(5, activation="softmax"))

model.layers[0].set\_weights([embedding\_matrix])

model.layers[0].trainable = False



# Preprocessing

Usunięcie znaków specjalnych, przystankowych

Usunięcie stop wordsów

Usunięcie liczb

Usunięcie odnośników

Usunięcie znaków nowej linii

Usunięcie wielokrotnych spacji (występujących po sobie)

Podział danych na zbiór treningowy oraz testowy w punkcie niedalekiej przeszłości

Tokenizacja danych i utworzenie sekwencji

# Model 2

model\_m2 = Sequential()

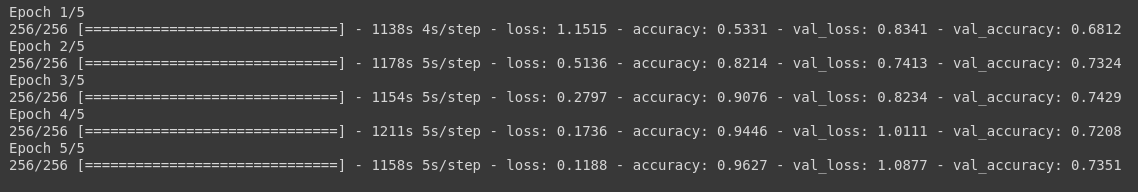
model\_m2.add(Embedding(MAX\_NB\_WORDS, EMBEDDING\_DIM, input\_length=X.shape[1]))

model\_m2.add(SpatialDropout1D(0.2))

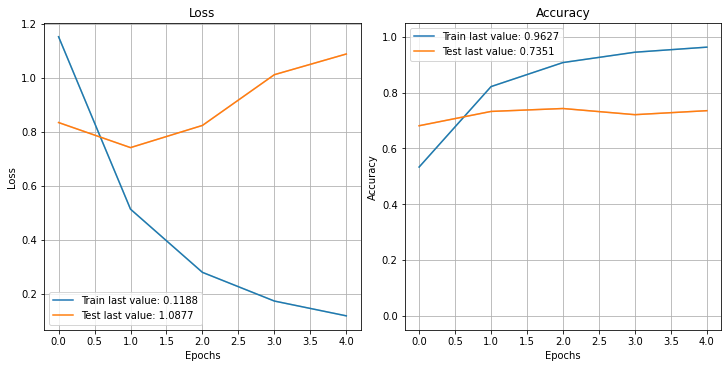
model\_m2.add(LSTM(100, dropout=0.2, recurrent\_dropout=0.2))

model\_m2.add(Dense(5, activation='softmax'))

model\_m2.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])



poprawa modelu jakości modelu w stosunku do pierwszego modelu



## Model 3

model\_m3 = Sequential()

model\_m3.add(Embedding(MAX\_NB\_WORDS, EMBEDDING\_DIM, input\_length=X.shape[1]))

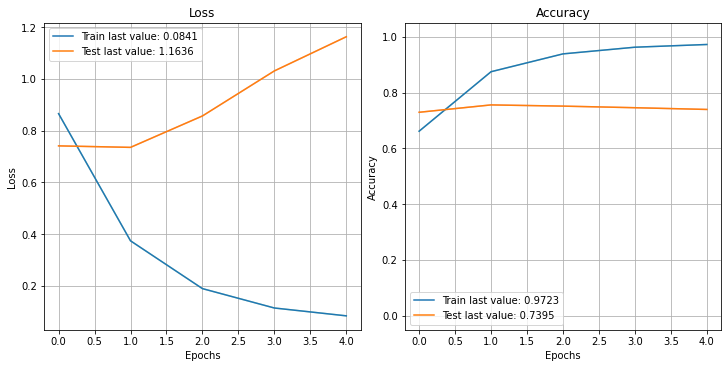
model\_m3.add(LSTM(100, dropout=0.2, recurrent\_dropout=0.2))

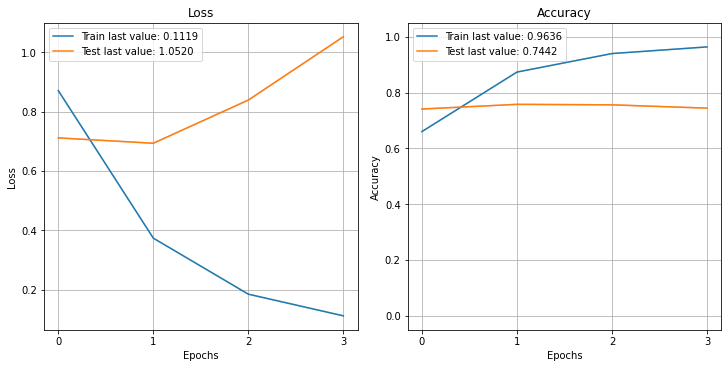
model\_m3.add(Dropout(0.2))

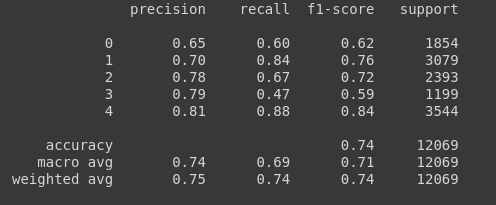
model\_m3.add(Dense(5, activation='softmax'))

model\_m3.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

Porównanie modelu z różnych random statem







# Model 4

model\_m4 = Sequential()

model\_m4.add(Embedding(MAX\_NB\_WORDS+1, EMBEDDING\_DIM, input\_length=X.shape[1]))

model\_m4.add(LSTM(128, dropout=0.2, recurrent\_dropout=0.2, return\_sequences=True))

model\_m4.add(Dropout(0.2))

model\_m4.add(LSTM(64, dropout=0.2, recurrent\_dropout=0.2))

model\_m4.add(Dropout(0.2))

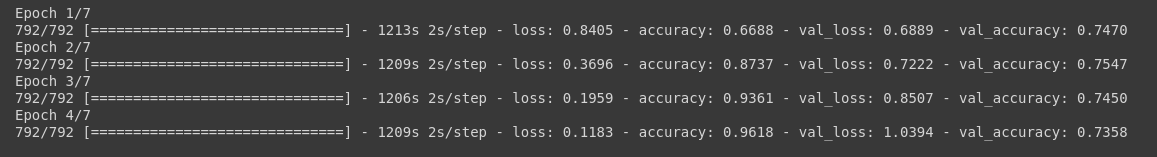
model\_m4.add(Dense(5, activation='softmax'))

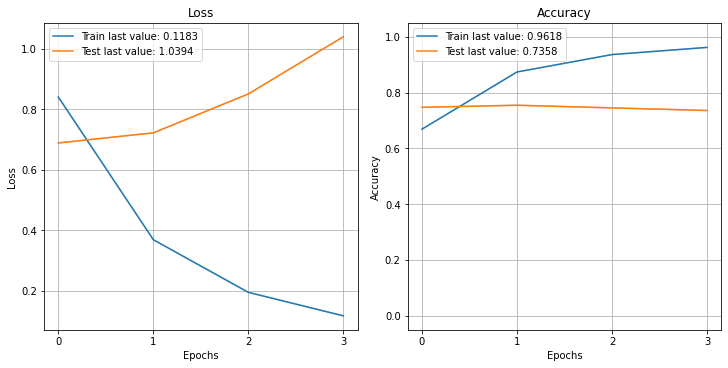
# model\_m4.layers[0].set\_weights([embedding\_matrix])

# model\_m4.layers[0].trainable = False

model\_m4.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

Model składający się z dwóch warstw LSTM





# Model 5

model\_m5 = Sequential()

model\_m5.add(Embedding(MAX\_NB\_WORDS+1, EMBEDDING\_DIM, input\_length=X.shape[1]))

model\_m5.add(GRU(128, dropout=0.2, recurrent\_dropout=0.2, return\_sequences=True))

model\_m5.add(Dropout(rate=0.2))

model\_m5.add(GRU(64, dropout=0.2, recurrent\_dropout=0.2, return\_sequences=True))

model\_m5.add(Dropout(rate=0.2))

model\_m5.add(GRU(32, dropout=0.2, recurrent\_dropout=0.2, return\_sequences=True))

model\_m5.add(Dropout(rate=0.2))

model\_m5.add(GRU(16, dropout=0.2, recurrent\_dropout=0.2))

model\_m5.add(Dropout(rate=0.2))

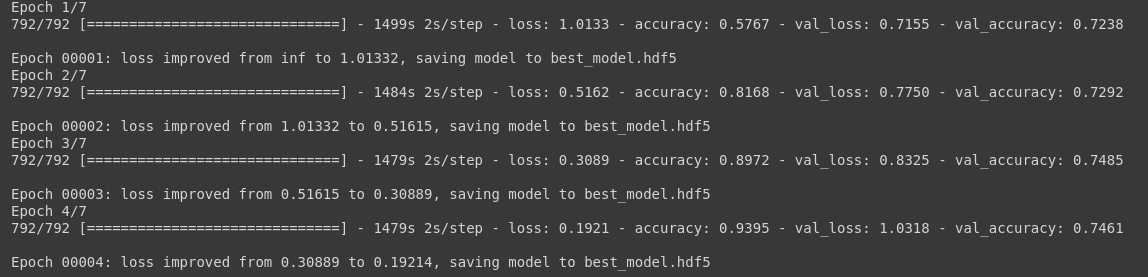
model\_m5.add(Flatten())

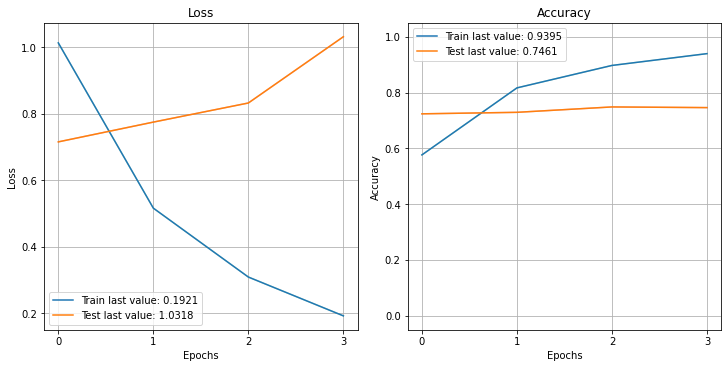
model\_m5.add(Dense(125, activation="relu"))

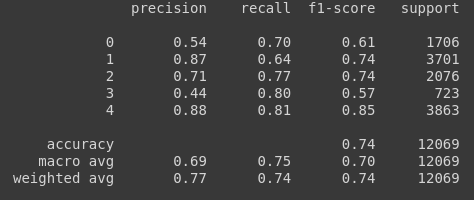
model\_m5.add(Dense(25, activation="relu"))

model\_m5.add(Dense(5, activation="softmax"))

Model bardzo długo się uczył – nie poprawiło to zbytnio ostatecznego wyniku mimo skomplikowania modelu. Wynik prawie jak nie najlepszy.







# Wczytanie modelu

Model nr 5 po wczytaniu osiągnął następujące wyniki

