Τεχνητή Νοημοσύνη 2 Εργασία 3

Βασίλειος Βασιλάκης A.M: 1115201800018

(1) Bidirectional Stacked RNN with LSTM/GRU cells for sentiment classification

Preprocessing:
Για τα μοντέλα χρησιμοποιήσαμε τα pre trained word embeddings του GloVe, και συγκεκριμένα το "glove.6B.200d.txt". Αρχικά εξάγουμε τα word embeddings, φτιάχνοντας τον embeddings array και το word-to-index dictionary. Το word-to-index dictionary, δίνει σε κάθε λέξη του vocabulary ένα μοναδικό index, και απλά αποθηκεύει αυτήν την αντιστοιχία ανάμεσα σε όλες τις λέξεις του vocabulary και στα indices τους. Στο index 0, έχουμε αντιστοιχίσει την κενή λέξη, η οποία δεν εμφανίζεται στο vocabulary, και απλά την προσθέσαμε γιατί μας χρησιμεύει στη μετατροπή των twitter sentences σε index sequences και στο padding των sequences που θα πούμε παρακάτω. Ο embeddings array είναι απλά ο πίνακας που στη γραμμή i έχει αποθηκεύσει το word embedding vector της λέξης που αντιστοιχεί στο index i του word-to-index dictionary. Στο index 0, έχουμε δώσει το τετριμμένο word embedding με μόνο μηδενικά.

Οι προτάσεις/tweets περνάνε από το data cleaning που χρησιμοποιούσαμε και στις προηγούμενες εργασίες.

Στη συνέχεια, όπως αναφέραμε, πρέπει με κάποιο τρόπο να μετατρέψουμε την κάθε πρόταση/tweet σε μια ακολουθία αριθμών, την οποία θα περάσουμε σαν είσοδο στα μοντέλα μας. Αυτό πλέον δεν θα το κάνουμε με τον αφελή τρόπο του averaging των embedding vectors κάθε πρότασης, αλλά με την μετατροπή κάθε πρότασης σε μια ακολουθία ακεραίων/δεικτών. Και η ακολουθία αυτή προκύπτει απλά αντικαθιστώντας κάθε λέξη της πρότασης, με τον δείκτη που της αντιστοιχεί στο word-to-index dictionary. Αν κάποια λέξη δεν υπάρχει στο vocabulary και κατ'επέκταση στο word-to-index dictionary τότε αντιστοιχίζεται στην κενή λέξη, δηλαδή στο index 0. Επίσης, επειδή το μήκος των ακολουθιών σε κάθε batch πρέπει να είναι σταθερό, επιλέξαμε κάθε προκύπτουσα ακολουθία να έχει ένα προκαθορισμένο σταθερό μήκος, το οποίο θα είναι αρκετά μεγάλο ώστε να μην χάνουμε πολλή πληροφορία, αλλά και αρκετά μικρό ώστε να μην αργεί υπερβολικά το training. Επειδή το μέγιστο πλήθος λέξεων που έχει κάποια πρόταση στο training set ήταν 39, επιλέξαμε αυτό το σταθερό μήκος να είναι 25. Συνεπώς,

οι ακολουθίες των προτάσεων που έχουν πλήθος λέξεων μεγαλύτερο από 25, περιορίζονται στους δείκτες των πρώτων 25 λέξεων (χάνοντας το sentiment των υπόλοιπων λέξεων), ενώ οι ακολουθίες των προτάσεων που έχουν πλήθος λέξεων μικρότερο από 25, γίνονται pad με τον μηδενικό δείκτη (δηλαδή συμπληρώνονται με το sentiment της κενής λέξης). Η συνάρτηση που κάνει αυτή τη δουλειά είναι η sentences_to_sequences στο κελί με τίτλο 'Useful Functions'

Στο κελί με τίτλο: 'Useful Functions', βρίσκονται επίσης χρήσιμες συναρτήσεις για data cleaning, εκτύπωση classification scores, plotting learning curves, plotting roc curves και μια γενική συνάρτηση για training των μοντέλων. Αυτές είναι πάνω κάτω ίδιες με τις προηγούμενες εργασίες. Μόνο η συνάρτηση **execute_training** που εκτελεί το training των μοντέλων έχει δύο σημαντικές αλλαγές:

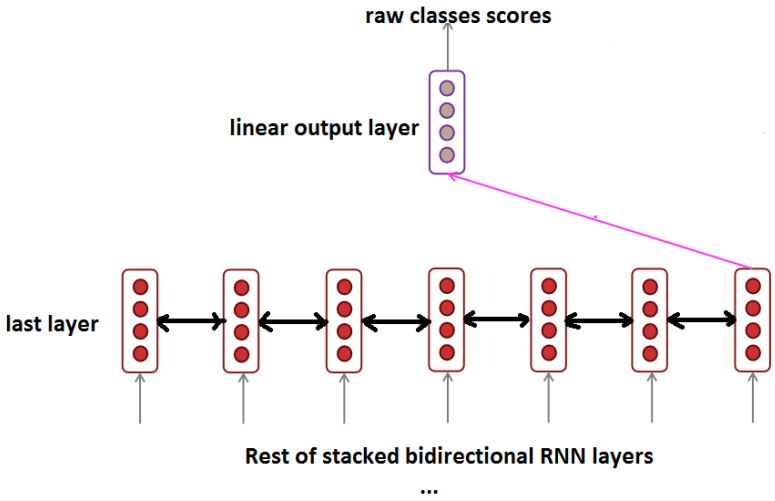
- Η πρώτη είναι ότι προσθέσαμε **gradient clipping** με threshold 2.0, για να περιορίσουμε (σε περίπτωση που εμφανίζεται) το φαινόμενο των exploding gradients.
- Η δεύτερη είναι ότι επειδή τα RNN-LSTM/GRU μοντέλα παρατηρήθηκε ότι μαθαίνουν αρκετά πιο γρήγορα, σε πολύ λιγότερες εποχές από τι τα μοντέλα των προηγούμενων εργασίων, και άρα ο κίνδυνος για overfitting είναι μεγαλύτερος, προσθέσαμε ένα είδος **early stopping** regularization. Συγκεκριμένα, κρατάμε το ελάχιστο validation loss μέχρι στιγμής, και αν αυτό δεν βελτιωθεί για 3 συνεχόμενες εποχές, τότε τερματίζουμε πρόωρα το training για να προλάβουμε το overfit στην αρχή του.

Experiments:
Αρχικά ορίσαμε την wrapper κλάση Classifier, η οποία ορίζει ένα μοντέλο με την εξής αρχιτεκτονική (η είσοδος και έξοδος αναφέρονται σε tensor shapes):

- ένα non-trainable embedding layer σαν πρώτο layer εισόδου
  ΕΙΣΟΔΟΣ : (B, S). ΕΞΟΔΟΣ : (B, S, E)
- ένα bidirectional stacked RNN με είτε lstm είτε gru cells
  ΕΙΣΟΔΟΣ : (B, S, E). ΕΞΟΔΟΣ : (B, S, 2H).
- ένα linear output layer που παράγει τα raw scores για καθε τις 3 κλάσεις
  ΕΙΣΟΔΟΣ : (B, 1, 2H). ΕΞΟΔΟΣ : (B,3)

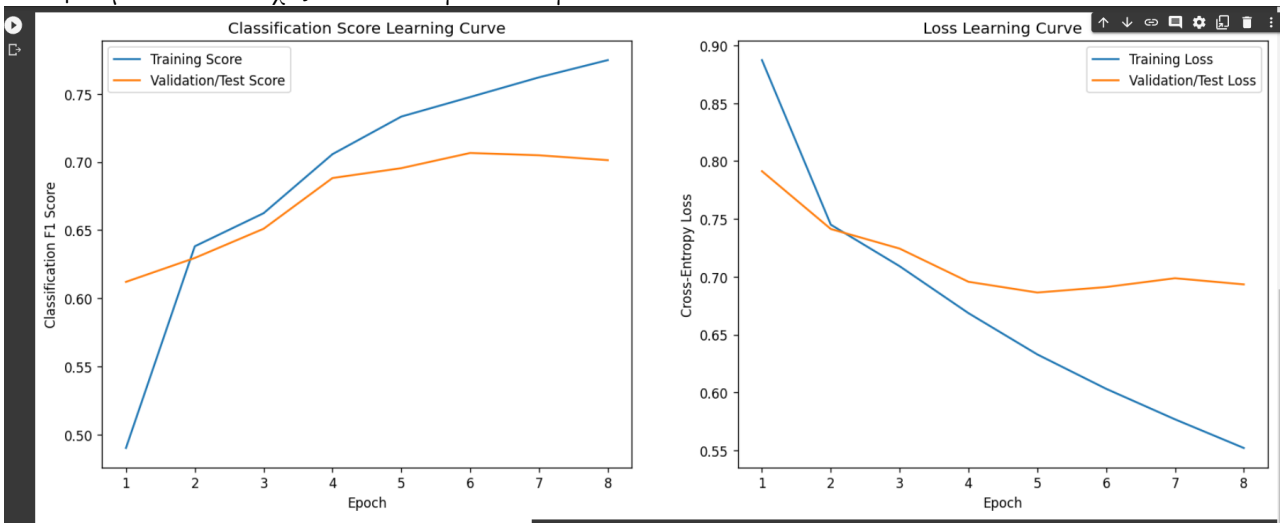
όπου B = batch_size , S = sequence_length, E = embeddings_dim , H = num_hidden

Το 2H προκύπτει από το ότι το stacked RNN είναι bidirectional άρα το output ενός hidden state είναι τα vectors και των δύο κατευθύνσεων ενωμένα σε ένα με διπλάσια διάσταση (2H). Επίσης να αναφέρουμε ότι η είσοδος στο linear output layer είναι μόνο (B, 1, 2H) παρότι η έξοδος του RNN είναι (B, S, 2H). Αυτό διότι χρησιμοποιούμε μόνο την έξοδο του τελευταίου/δεξιότερου hidden state του ανώτερου layer. 'Εχουμε δηλαδή την εξής εικόνα:

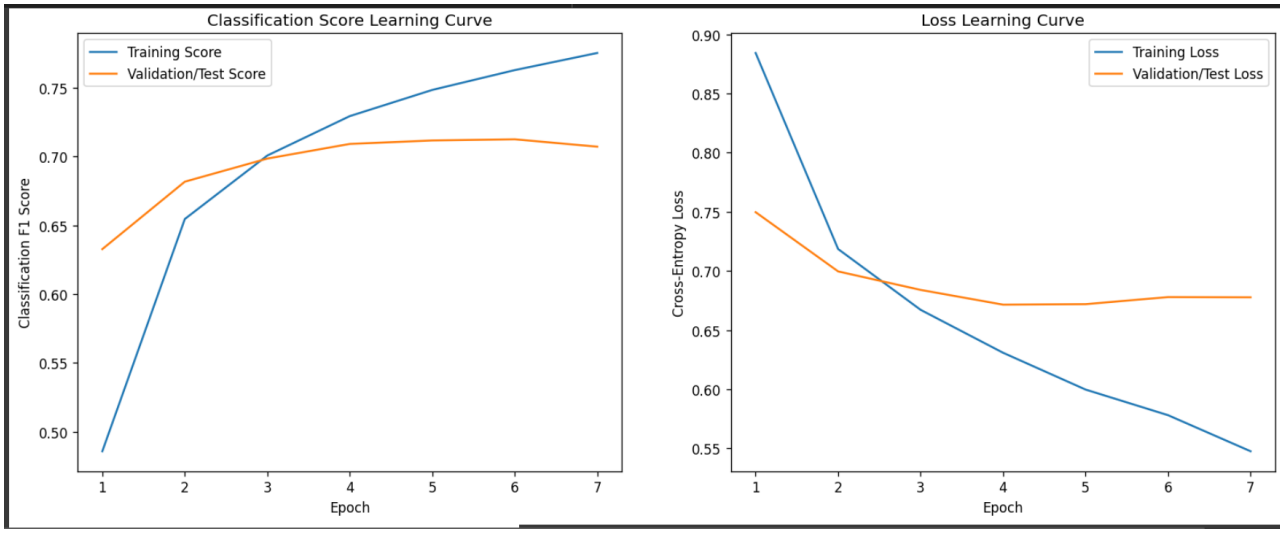


Αυτό το αναφέρουμε, για να έρθει σε αντιδιαστολή αργότερα όταν θα προσθέσουμε attention στο μοντέλο.

Το πρώτο μοντέλο που δοκιμάσαμε λοιπόν είναι ένα απλό bidirectional rnn με lstm cells και ένα μόνο layer, με num_hidden = 50. Το learning rate ίσο με 1e-3 = 0.001. Από τα αποτελέσματα του μοντέλου βλέπουμε ότι παρότι το μοντέλο έχει αρκετά καλά roc curves και πετυχαίνει αρκετά καλό validation σκορ (περίπου 0.70-0.71, ανάλογα και με την εκτέλεση), δεν έχει αρκετά σταθερή συμπεριφορά όπως θα θέλαμε, με το validation loss να παρουσιάζει συχνά μικρές αυξομειώσεις σε διαδοχικές εποχές, οι καμπύλες του training και validation loss/score να απομακρύνονται, και πολλές φορές να αρχίζει overfitting και να έχουμε early stopping σε λιγότερο από 10 εποχές. Ενδεικτική εκτέλεση:



Το δεύτερο μοντέλο που δοκιμάσαμε είναι πανομοιότυπο με το πρώτο, μόνο που χρησιμοποιεί GRU cells αντί για lstm cells. Η συμπεριφορά του μοντέλου ήταν πάλι η ίδια και ότι ειπώθηκε για το πρώτο μοντέλο μπορεί να ειπωθεί και σε αυτό το μοντέλο. Το σκορ τόσο στο training όσο και στο validation set, παρατηρήθηκε ότι ήταν συγκρίσιμο με το πρώτο lstm μοντέλο. Ενδεικτική εκτέλεση του δεύτερου μοντέλου:



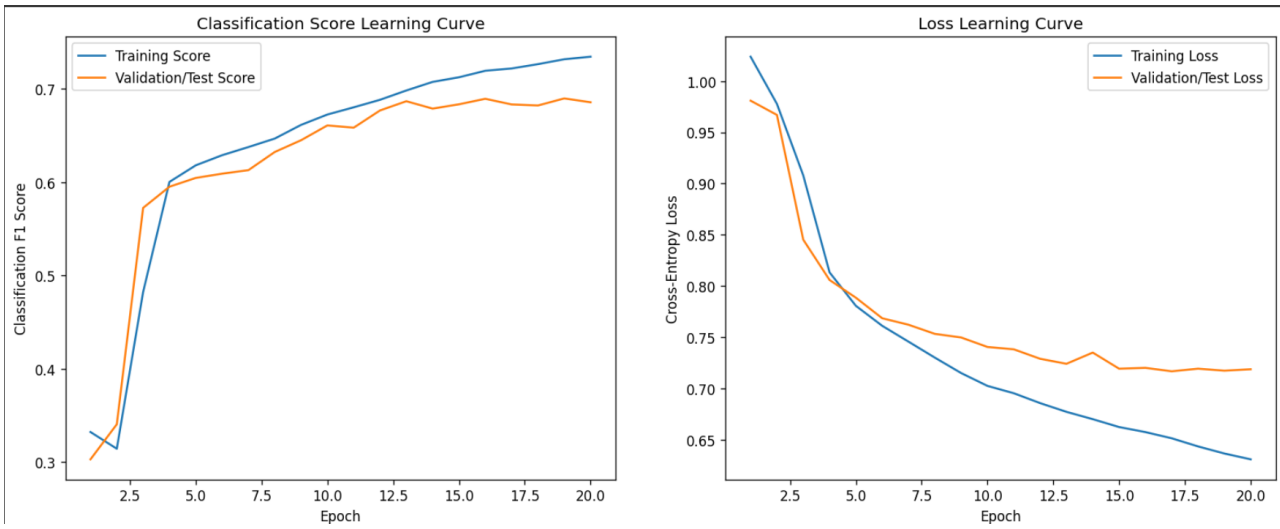
Γενικά να αναφέρουμε σε αυτό το σημείο ότι παρατηρήθηκε ότι τα μοντέλα που χρησιμοποιούσαν GRU cells αντί για lstm, αφενός είχαν λίγο γρηγορότερο training speed, κάτι λογικό αφού τα lstm μοντέλα έχουν αρκετά παραπάνω παραμέτρους, αφετέρου είχαν λίγο καλύτερη συμπεριφορά και πετύχαιναν λίγο καλύτερο σκορ στο δεδομένο μικρό μας dataset. Ίσως σε μεγαλύτερο dataset το lstm να ήταν καλύτερο.

Στη συνέχεια, με σκοπό να κάνουμε το μοντέλο πιο σταθερό και να μειώσουμε το overfitting, μειώσαμε κατ'αρχήν το num_hidden από 50 σε 20, και το πιο σημαντικό μειώσαμε το learning rate σε 3e-4. Λόγω της μείωσης του learning rate, αυξήσαμε και τον αριθμό των εποχών του training. Τέλος, πειραματιστήκαμε με το να προσθέσουμε ένα επιπλέον layer.

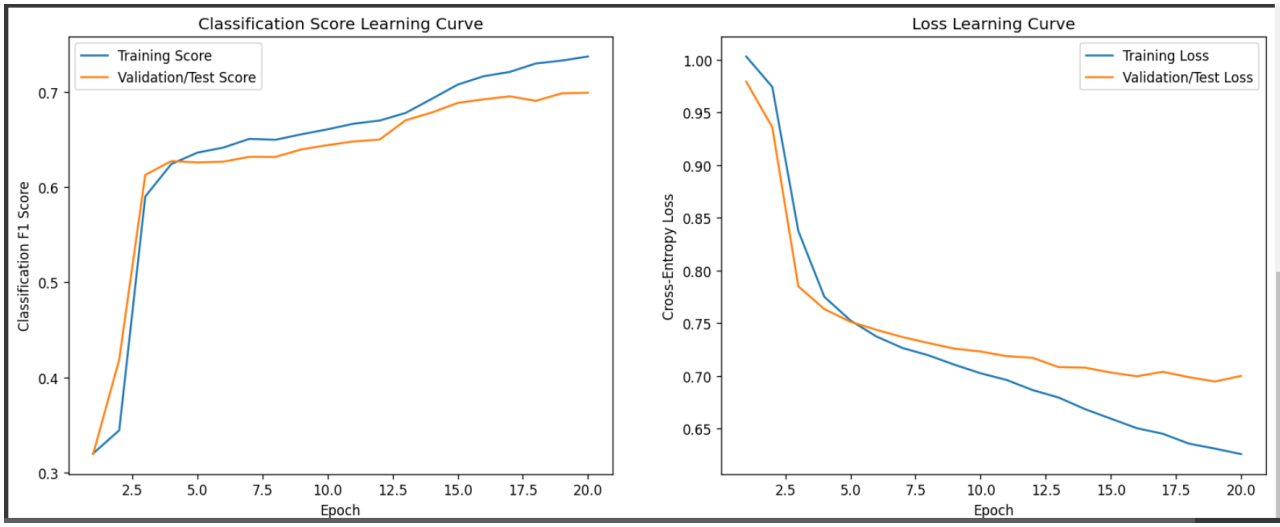
Έτσι φτάνουμε στο τρίτο και τέταρτο μοντέλο, με τις νέες παραμέτρους που προαναφέραμε, και με lstm και gru cells αντίστοιχα. 'Οπως βλέπουμε έχουμε αρκετά πιο σταθερή συμπεριφορά, οι καμπύλες τόσο του training score vs validation score όσο και του training loss vs validation loss είναι πιο κοντά μεταξύ τους και η καμπύλη του validation loss μοιάζει να επιπεδοποιείται στο τέλος και να μην αυξάνεται όπως πριν, άρα έχει μειωθεί και το overfitting. Το μοντέλο με gru cells

τα πηγαίνει λίγο καλύτερα από το μοντέλο με lstm cells, τόσο ως προς το σκορ του classification, όσο ως προς το σκορ των roc curves. Το μόνο αρνητικό είναι ότι μειώθηκε λίγο το validation score σε σχέση με τα πρώτα δύο μοντέλα, και στο μοντέλο με lstm είναι γύρω στο 0.66-0.68 ενώ στο μοντέλο με gru είναι στο 0.69-0.70.

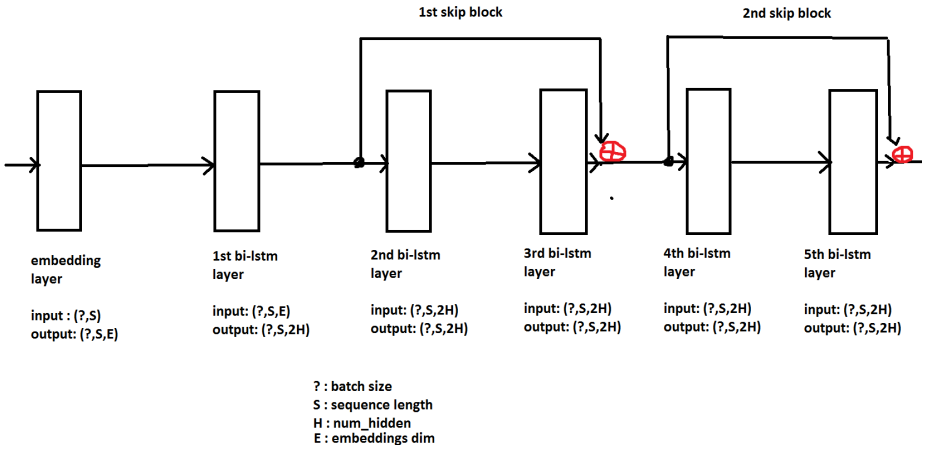
Ενδεικτική εκτέλεση για το μοντέλο με lstm cells (μοντέλο 3):



Ενδεικτική εκτέλεση για το μοντέλο με gru cells (μοντέλο 4):



Στη συνέχεια, προσθέσαμε skip connections στο τέταρτο μοντέλο , για να πειραματιστούμε με λίγο βαθύτερα δίκτυα RNN. Τα skip connections, πέρα από το ότι βοηθάνε και με το πρόβλημα των vanishing gradients, διευκολύνουν τα deep RNN να συνεχίσουν να μαθαίνουν παρά την αύξηση των layers. Κάτι που θεωρητικά θα έπρεπε να συμβαίνει ούτως ή άλλως, αλλά στη πράξη δεν συμβαίνει πάντα. Η αρχιτεκτονική του μοντέλου, μαζί με τα skip connections που υλοποιήσαμε, γραφικά είναι ως εξής:



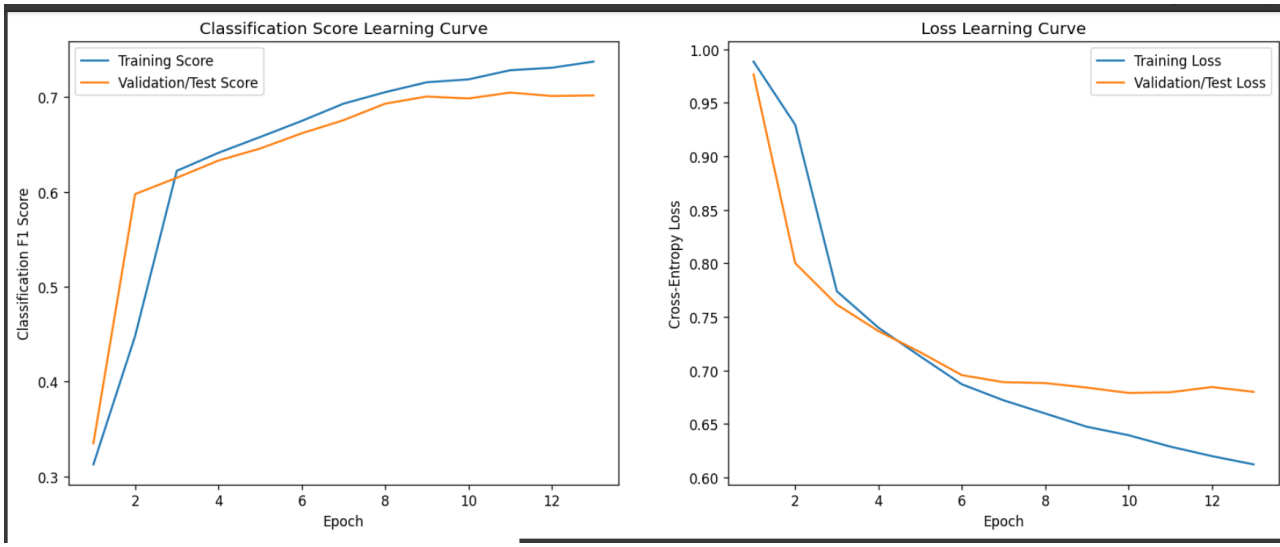
Στη φωτογραφία φαίνεται ένα snapshot της αρχιτεκτονικής. Ενδεχομένως να ακολουθούν και άλλα skip blocks και στο τέλος το linear output layer όπως και πριν.

Ουσιαστικά, αυτό που γίνεται είναι ότι έχουμε μετά το embedding input layer, ένα πρώτο bidirectional lstm/gru layer, το οποίο λειτουργεί και σαν "διορθωτής διαστάσεων", καθώς μετατρέπει την είσοδό του, σε μία έξοδο με τις επιθυμητές διαστάσεις ώστε να μπορεί να γίνει μετά η πρόσθεση στα skip blocks. Μετά το πρώτο αυτό lstm/gru layer, έρχεται μια ακολουθία από skip blocks, όπου κάθε skip block είναι δύο lstm/gru layers στη σειρά, όπου η είσοδος του πρώτου προστίθεται στην έξοδο του δεύτερου, ώστε να μεταφέρεται με αυτόν τον τρόπο πληροφορία άθικτη βαθύτερα στο δίκτυο. Ο κώδικας στην κλάση SkipClassifier υλοποιεί ακριβώς αυτήν την αρχιτεκτονική.

'Αρα, για πειραματισμό, πήραμε το 4ο μοντέλο, του βάλαμε skip connections και αυξήσαμε τον αριθμό των layers σε 5. Σαν αποτέλεσμα είχαμε ένα μοντέλο το οποίο είναι πιο χρονοβόρο και resource hungry από άποψη training, κάτι λογικό αφού η προσθήκη επιπέδων αυξάνει δραματικά το πλήθος των παραμέτρων προς μάθηση. Αλλά το θετικό είναι ότι παρατηρήθηκε ότι έχουμε λίγο καλυτερο validation σκόρ (γύρω στο 70-71) σε σχέση με πριν, κάτι που αναμέναμε, και παράλληλα

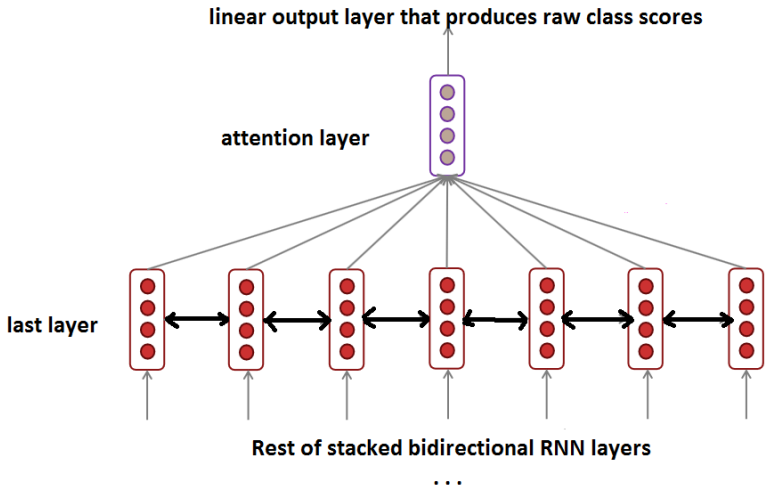
οι καμπύλες του loss και του f1 score εξακολουθούν να είναι κοντά μεταξύ τους, εξακολουθούμε να έχουμε σχετικά καλό fit, με μια μικρή τάση για overfitting προς το τέλος όμως σε μερικές εκτελέσεις.

Ενδεικτική εκτέλεση:



(2) Adding attention to best model
Σαν καλύτερο μοντέλο επιλέχθηκε το 4ο μοντέλο, χωρίς τα skip connections.

Αυτό που γίνεται, είναι ότι πλέον δεν χρησιμοποιούμε μόνο το output του τελευταίου-δεξιότερου hidden state του ανώτερου layer, αλλά το output ολόκληρου του ανώτερου layer, από όλα τα hidden states. 'Εχουμε δηλαδή διαισθητικά την εξής εικόνα:



Το attention layer βέβαια είναι λίγο πιο περίπλοκο μαθηματικά.
Η αρχιτεκτονική του μοντέλου, που υλοποιείται στην κλάση AttentionClassifier είναι η εξής:

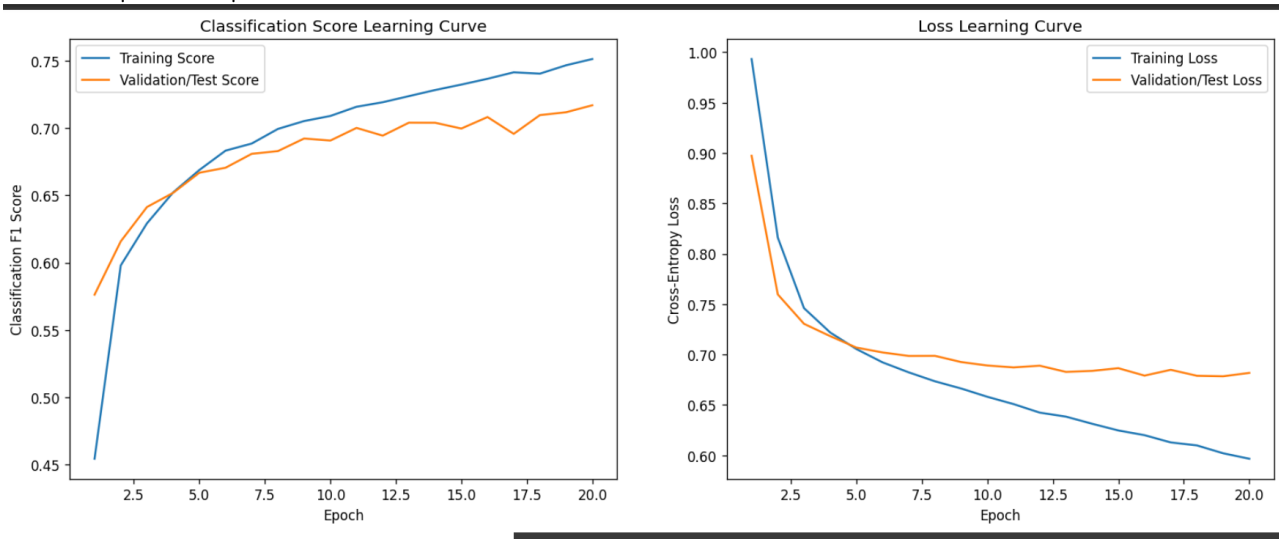
- πάλι ένα non-trainable embedding layer σαν πρώτο layer εισόδου
  ΕΙΣΟΔΟΣ : (B, S). ΕΞΟΔΟΣ : (B, S, E)
- πάλι ένα bidirectional stacked RNN με είτε lstm είτε gru cells
  ΕΙΣΟΔΟΣ : (B, S, E). ΕΞΟΔΟΣ : (B, S, 2H).
- ένα linear layer με όνομα attention layer, του οποίου τα trainable weights πολλαπλασιάζονται με το output του RNN, και προκύπτουν σαν έξοδος τα attention weights (ένα weight per hidden state)
  ΕΙΣΟΔΟΣ : (B, S, 2H). ΕΞΟΔΟΣ : (B, S, 1)
- Η έξοδος του attention layer με shape (B, S, 1) περνάει μετά από μια tanh, την activation του attention layer, και μετά από μια softmax, ώστε τα attention weights να αθροίζουν σε 1.
  ΕΙΣΟΔΟΣ : (B, S, 1). ΕΞΟΔΟΣ : (B, S, 1)
- Τέλος τα S attention weights πολλαπλασιάζονται element wise με τα S output vectors των hidden states του τελευταίου layer του RNN, και τα S vectors που προκύπτουν προστίθενται
  ΕΞΟΔΟΣ : (B, 1, 2H)
- ένα linear output layer που παράγει τα raw scores για καθε τις 3 κλάσεις
  ΕΙΣΟΔΟΣ : (B, 1, 2H). ΕΞΟΔΟΣ : (B,3)

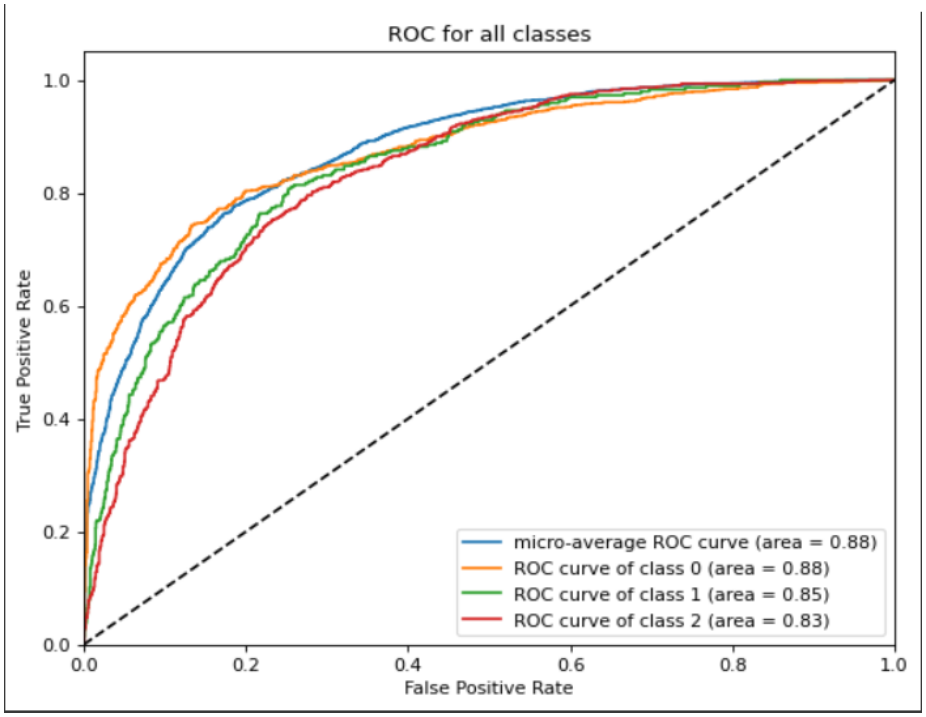
όπου B = batch_size , S = sequence_length, E = embeddings_dim , H = num_hidden

Ως προς τα αποτελέσματα, το μοντέλο μετά την προσθήκη του attention, παρουσιάζει

μια μικρή αλλά εμφανή βελτίωση. Οι καμπύλες του training loss vs validation loss αλλά και οι καμπύλες του training score vs validation score είναι κοντά μεταξύ τους, και η καμπύλη του validation loss φαίνεται να επιπεδοποιείται στο τέλος χωρίς τάση προς αύξηση, εξακολουθούμε δηλαδή να έχουμε αρκετά καλό fit. Το classification score επίσης βελτιώνεται, τόσο συνολικά όσο και ανά κλάση, και στο validation set από 0.69 που ήταν πριν, μετά την προσθήκη του attention είναι σταθερά πάνω από 0.70, και σε μερικές εκτελέσεις φτάνει μέχρι και το 0.71. Τέλος οι roc curves είναι επίσης μετατοπισμένες λίγο παραπάνω προς τα πάνω αριστερά, και το εμβαδόν τους είναι σε όλες τις κλάσεις μεγαλύτερο ή ίσο σε σχέση με πριν.

Ενδεικτική εκτέλεση:





Οι βελτιώσεις ήταν αναμενόμενες, βέβαια, αφού πλέον δεν περιοριζόμαστε μόνο σε μία αλλά χρησιμοποιούμε όλες τις hidden states του ανώτερου layer του RNN, και επίσης προσθέτουμε και weights προς εκπαίδευση στο attention layer, τα οποία ανάλογα με την πρόταση στην είσοδο, θα παράγουν και διαφορετικά attention weights για κάθε λέξη/hidden state τα οποία μας υποδηλώνουν σε κάποιο βαθμό το context των λέξεων και βοηθούν το sentiment analysis.

Τέλος παρατηρούμε, ότι το καλύτερο μας μοντέλο + attention καταφέρνει να ξεπεράσει τόσο σε classification score (συνολικό αλλά και ανά κλάση) όσο και σε roc curve score, το καλύτερο μοντέλο της δεύτερης εργασίας, αν και για πολύ λίγο, ενώ έχει συγκρίσιμα αποτελέσματα με το καλύτερο μοντέλο της πρώτης. Βέβαια με περισσότερα δεδομένα, η απόκλιση θα ήταν ακόμα μεγαλύτερη.

Note 1: Για τις μετρικές precision, recall, f1 score χρησιμοποιήθηκε weighted averaging, καθώς οι κλάσεις εμφανίζονται με διαφορετικές συχνότητες στα data. Επίσης τυπώνονται και τα precison, recall, f1 score ανά κλάση. Για την κλάση 1, τα scores είναι σημαντικά μικρότερα σε σχέση με τις άλλες 2, κάτι λογικό αφού έχει αρκετά μικρότερη συχνότητα εμφάνισης στο dataset. Αυτό είναι επίσης κάτι που εμποδίζει τα μοντέλα μας να πάνε σε καλύτερο σκορ.

Note 2: Το αργότερο μοντέλο είναι το βαθύ με τα skip connections, που παίρνει γύρω στα 8 λεπτά για εκπαίδευση σε CPU. Μπορείτε βέβαια να τρέξετε το colab

με GPU για επιτάχυνση.

Note 3: Τα μοντέλα ενδέχεται να έχουν διαφορετικά αποτελέσματα ανάμεσα σε διαφορετικά sessions εκτέλεσης. Παρά τα seeds, δεν είναι πλήρως ντετερμινιστικά. Οι παρατηρήσεις του report όμως είναι μια γενική/συνολική εικόνα του τι παρατηρήθηκε μετά από αρκετές εκτελέσεις των ίδιων μοντέλων.

Note 4: Το τελικό μοντέλο βρίσκεται στο τελευταίο κελί του colab με επικεφαλίδα final model. Θα πρέπει να εκτελεστούν εκτός από το τελευταίο κελί με το τελικό μοντέλο προφανώς, τουλάχιστον τα 3 πρώτα κελιά για να γίνουν τα imports το κατέβασμα των word embeddings, και ο ορισμός των συναρτήσεων

Note 5: Το φόρτωμα του test set γίνεται απλά αντικαθιστώντας το όρισμα στην **pd.read_csv("./sample_data/vaccine_validation_set.csv")** με το μονοπάτι του test set (έχουμε θεωρήσει ότι τα train, validation, test set τοποθετούνται στον κατάλογο ./sample_data)