Επεξεργασία Φυσικής Γλώσσας – 2022/20223 – Ο.Π.Α

**Εργασία 4**

Φοίβος-Αστέριος Νταντάμης (f3312204 ) ([phivos93@yahoo.com](mailto:phivos93@yahoo.com))

**Άσκηση 1 notebook url: https://colab.research.google.com/drive/1BxGYsttx1HnUWId0sq-ztt68aDIyslsG?usp=sharing**

**Περιγραφή Dataset:**

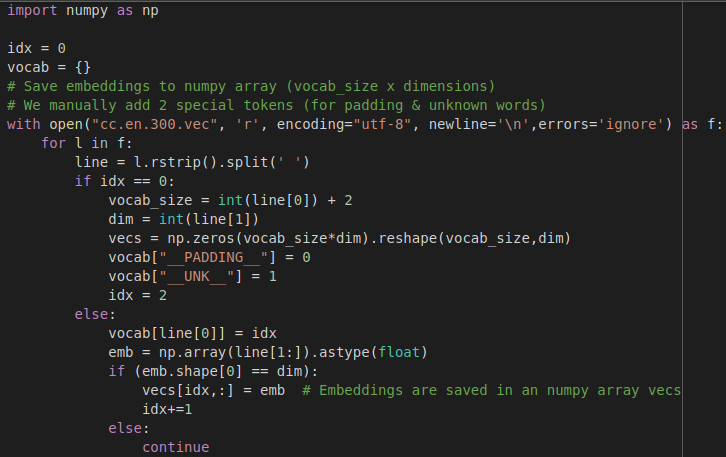
Το dataset είναι το ίδιο με τις δύο προηγούμενες ασκήσεις και έχει ήδη περιγραφεί εκεί.

**Word Embeddings:**

Μας ζητείται να αντιμετωπίσουμε το πρόβλημα του classification των κειμένων του dataset, αυτήν την φορά με την χρήση RNNs. Θα χρησιμοποιήσουμε για το σκοπό αυτόν έτοιμα word embeddings και πιο συγκεκριμένα τα fasttext word embeddings. Ξεκινάμε λοιπόν κατεβάζοντας το αρχείο που τα περιέχει

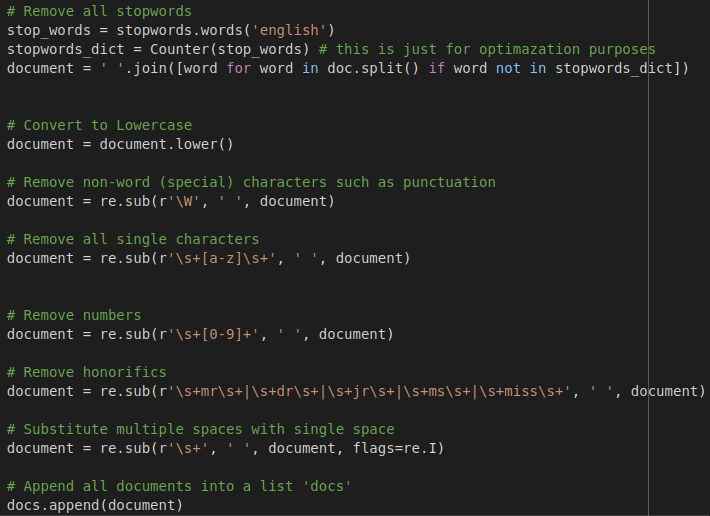


Για λόγους απόδοσης στη συνέχεια διαβάζουμε τα embeddings μία φορά και δημιουργούμε έναν δισδιάστατο πίνακα που περιέχει τα διανύσματα που τα περιγράφουν και ένα dictionary που αποθηκεύει τον αριθμό της γραμμής που έχει αποθηκευτεί το διάνυσμα κάθε λέξης, δηλαδή κρατάμε ένα πολύ απλό index. Προσθέτουμε δύο ειδικές λέξεις, το PADDING που θα χρησιμοποιηθεί για το “γέμισμα” των προτάσεων εισόδου που θα είναι πιο μικρές από το μέγεθος που θα ορίσουμε, όπως και το UNK που θα αντικαταστήσει τις λέξεις εκτός λεξικού.



**Dataset Preprocessing:**

To preprocessing έχει απλοποιηθεί σε σχέση με τις προηγούμενες ασκήσεις. Αυτό συνέβη διότι θέλουμε τελικά το dataset να περιέχει λέξεις οι οποίες μπορούν να απαντηθούν στο λεξικό, οπότε δεν τις αλλοιώσαμε.



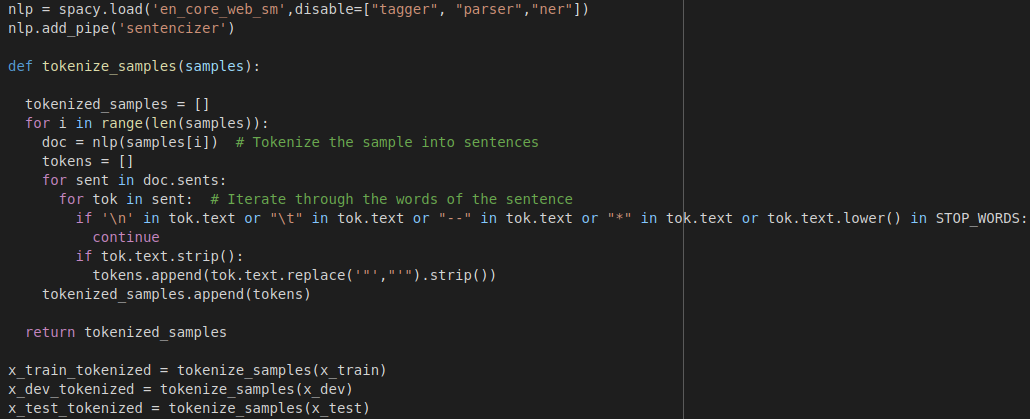
**Dataset Splitting:**

Χωρίζουμε τώρα το αρχικό dataset σε train, development και test δεδομένα με ποσοστά 70%, 20% και 10% αντίστοιχα.

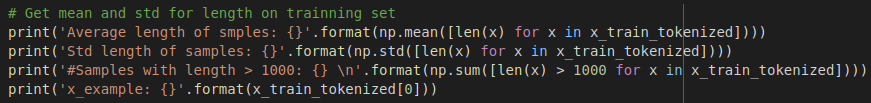


**Tokenization:**

Στη συνέχεια, σπάμε τα δεδομένα μας σε προτάσεις και μετατρέπουμε τις προτάσεις αυτές σε λίστες από tokens.

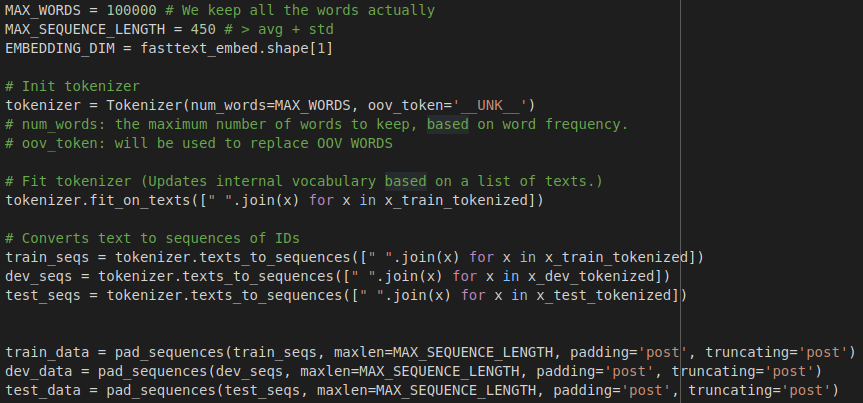


Υπολογίζουμε τώρα το μέσο μήκος, αλλά και την τυπική απόκλιση των μηκών όλων των προτάσεων στα train δεδομένα.



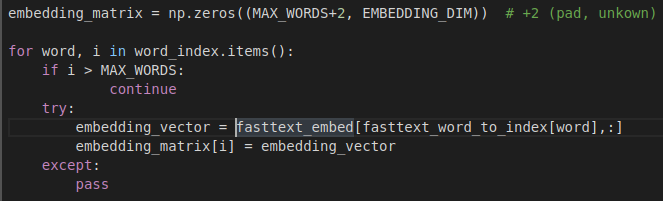
**Sequences:**

Στο επόμενο βήμα μετατρέπουμε τις προτάσεις του dataset μας σε ακολουθίες ακεραίων οι οποίοι αντιστοιχούν στο index των λέξεων στην αντίστοιχη θέση. Ταυτόχρονα δημιουργούμε το λεξικό μας που περιέχει μόνο τις λέξεις που συναντώνται στα train δεδομένα. Το μέγεθος μιας ακολουθίας ορίστηκε να είναι ίσο με 450 και ισούται περίπου με το άθροισμα του μέσο όρου και της τυπικής απόκλισης ώστε να αφήσει λίγες λέξεις εκτός σε κάποιες πιο μεγάλες προτάσεις, αλλά παράλληλα να μην χρειαστεί υπερβολικό padding στις μικρές.



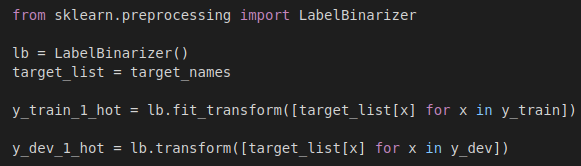
**Embedding matrix:**

Αρχικοποιούμε τον πίνακα με τα embeddings μας προσθέτοντας μόνο τις λέξεις που ανήκουν στο λεξικό που δημιουργήσαμε. Κρατάμε δηλαδή από το fasttext μόνο ό,τι μας χρειάζεται.



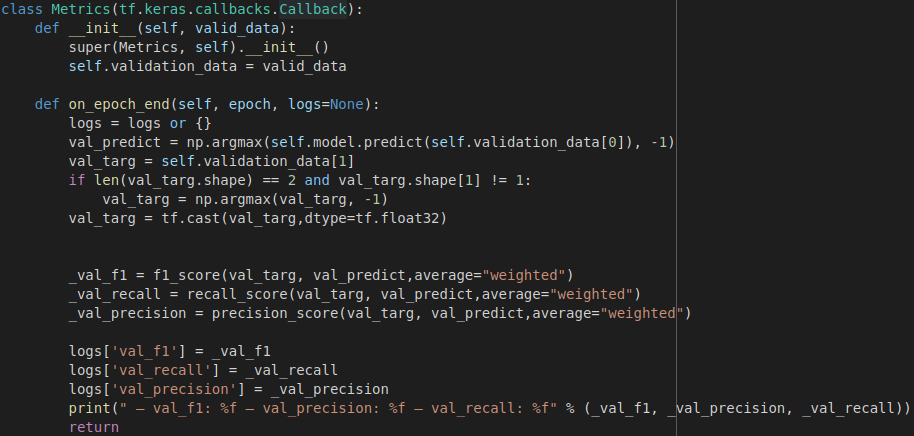
**1-hot vectors:**

Δημιουργούμε τα 1-hot vectors των δύο κλάσεων μας



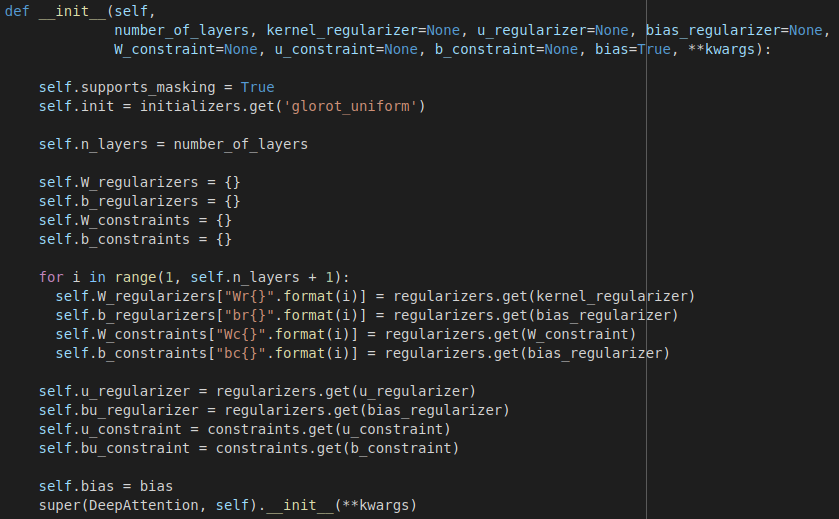
**Metrics Callback:**

Ορίζουμε όπως και την τελευταία φορά μία callback συνάρτηση που θα χρησιμοποιηθεί για την καταγραφή των μετρικών που μας ενδιαφέρουν στο τέλος κάθε εποχής.



**Deep attention layer:**

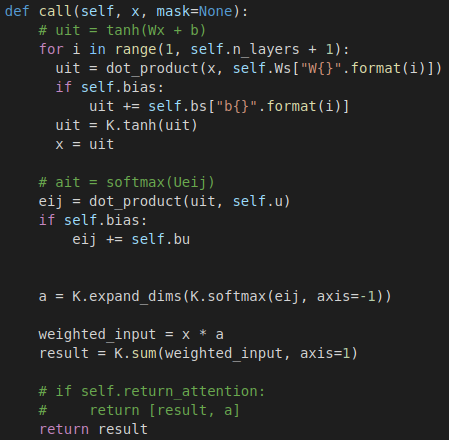
Ορίζουμε τώρα ένα custom keras layer που θα υλοποιεί το μηχανισμό του self attention. Παραμετροποιούμε την κλάση ως προς τον αριθμό των επιπέδων που θα δημιουργεί στο MLP που θα χρησιμοποιεί. Στον constructor λοιπόν δημιουργούμε dictionaries για τους regularizers και τα constraints για όλα τα W και b που θα χρειαστούν, ανάλογα με το πόσα επίπεδα θα δημιουργηθούν.



Μέσα στη μέθοδο build που χτίζει το layer μέσα σε ένα μοντέλο, αρχικοποιούνται όλοι οι πίνακες W και b, δηλαδή με άλλα λόγια αρχικοποιούνται τα βάρη όλων των νευρώνων για όλα τα επίπεδα του MLP.

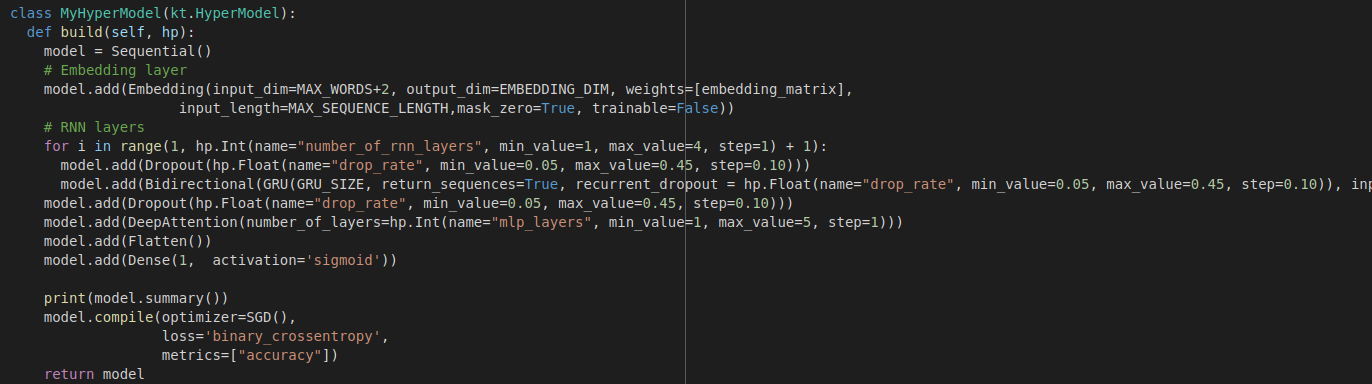


Στην μέθοδο call υλοποιούμε πλέον το layer και αφού κάθε πίνακας W αντιστοιχεί ουσιαστικά σε ένα επίπεδο ενός MLP, διαδίδουμε τα αποτελέσματα από το ένα επίπεδο στο επόμενο πολλαπλασιάζοντας με τη σειρά τα αποτελέσματα που προκύπτουν με τους πίνακες αυτούς.

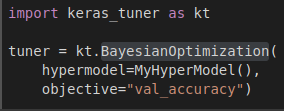
****

**Tuning:**

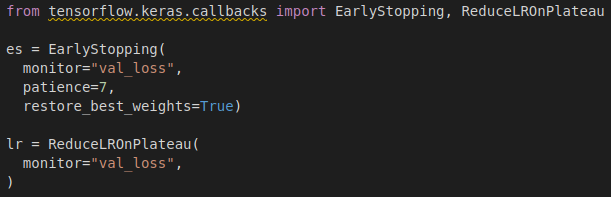
Προκειμένου να κάνουμε tune τις υπερπαραμέτρους που θέλουμε, αυτήν την φορά κάνουμε χρήση του keras tuner. Για να χρησιμοποιήσουμε οποιονδήποτε από τους διαθέσιμους tuners, πρέπει να φτιάξουμε αρχικά ένα HyperModel μέσα στο οποίο θα ορίζεται και ο χώρος αναζήτησης των υπερπαραμέτρων.



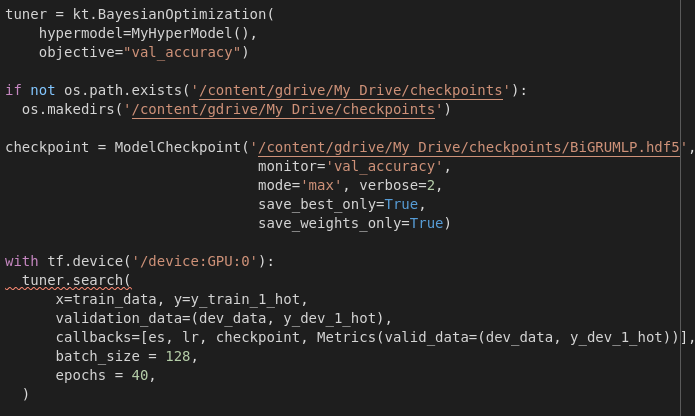
Μπορούμε τώρα το HyperModel αυτό να το περάσουμε σε κάποιον tuner και συγκεκριμένα στον BayesianOptimization.



Πριν ξεκινήσει η αναζήτηση, ορίζουμε δύο ακόμα callback συναρτήσεις. Την EarlyStopping που θα είναι υπεύθυνη για τον τερματισμό της αναζήτησης σε περιπτώσεις που τα αποτελέσματα δεν βελτιώνονται και την ReduceLROnPlateau που θα μπορεί να τροποποιεί το ρυθμό μάθησης στο τέλος κάθε εποχής.



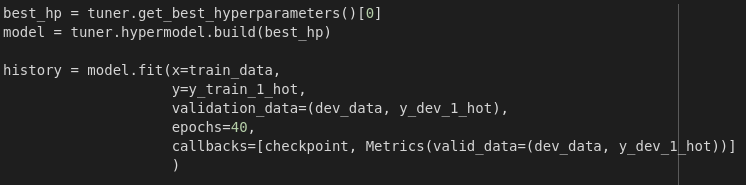
Στη συνέχεια με αξιοποίηση αυτήν την φορά και της GPU εκκινεί η αναζήτηση του βέλτιστου συνδυασμού των υπερπαραμέτρων.



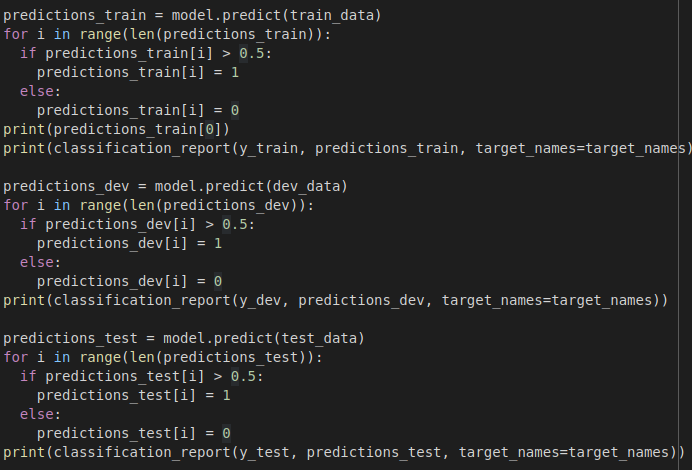
Σε αυτό το σημείο θα πρέπει να απολογηθούμε για την επιλογή ακόμα μια φορά όχι των συνολικά βέλτιστων παραμέτρων, αλλά φαίνεται αδύνατο να προλάβει να τελειώσει η αναζήτηση πριν μας διακόψει το collab ή περιορίσει την πρόσβαση στην gpu.

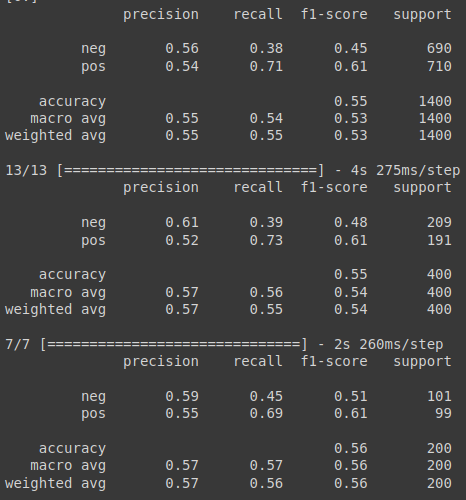
**Δημιουργία του τελικού μοντέλου και εκπαίδευσή του:**

Έχοντας κρατήσει τον καλύτερο συνδυασμό υπερπαραμέτρων, χτίζουμε το τελικό μοντέλο και το εκπαιδεύουμε πάνω στα δεδομένα εκπαίδευσης.



Εμφανίζoυμε τώρα τα reports με τις ζητούμενες μετρικές. Λείπει το auc, μιας και έπρεπε να περαστεί επίσης ως callback function για να υπολογίζεται, μα είναι αργά πλέον.





Τέλος, ζωγραφίζουμε τα διαγράμματα των acuracy και loss συναρτήσει των εποχών. Οι καμπύλες δεν έχουν προλάβει να συγκλίνουν, χρειαζόμασταν πιο πολλές εποχές. Ένα σημαντικό λάθος επίσης που έγινε είναι ότι ενώ ορίστηκε για early stopping ένα διάστημα μη βελτίωσης των 7 εποχών, για την προσαρμογή του learning rate η default τιμή του patience ήταν 10. Προφανώς θα έχουμε πάντα early stopping πριν προλάβει να αλλάξει το learning rate.

