ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία Φωνής και Φυσικής Γλώσσας
Χειμερινό Εξάμηνο 2020-2021

3η Εργαστηριακή Άσκηση

Φοιτητές: Λιακόπουλος Δημήτριος (03117109)
Μπεκρής Δημήτριος (03117116)

ΠΡΟΠΑΡΑΣΚΕΥΗ

Βήμα 1.1: Κωδικοποίηση Επισημειώσεων (Labels)

ΖΗΤΟΥΜΕΝΟ 1

Στο ερώτημα αυτό απαιτείται να δημιουργήσουμε labels για τις κλάσεις στις οποίες ανήκουν τα δεδομένα μας. Αρχικά, παρατηρούμε ότι ανάλογα το Dataset (MR ή Semeval2017A) που επιλέγουμε έχουμε διαφορετικό πλήθος κλάσεων. Στην περίπτωση του MR, παρατηρούμε ότι υπάρχουν δύο κλάσεις, αρνητική και θετική, ένω στην περίπτωση του Semeval2017A υπάρχουν τρεις διακριτές κλάσεις, αρνητική, ουδέτερη και θετική. Συνεπώς παρατηρούμε ότι για κάθε Dataset που μπορεί να δωθεί δεν υπάρχει προκαθορισμένο πλήθος κλάσεων. Έτσι για να δημιουργήσουμε τα αντιπροσωπεθτικά labels για κάθε κλάση, επιλέγεται ένας γενικος τρόπος μέσω της χρήσης του “LabelEncoder” της βιβλιοθήκης scikit-learn. Έτσι μπορούμε να αντιστοιχίσουμε την κάθε κλάση σε ένα μοναδικό label, ενώ παράλληλα αποθηκεύσουμε και το πλήθος των labels που δημιουργούνται για περαιτέρω χρήση.

Για το dataset MR, τα labels των δέκα πρώτων στοιχείων είνια τα ακόλουθα:

1. positive --> 1
2. positive --> 1
3. positive --> 1
4. positive --> 1
5. positive --> 1
6. positive --> 1
7. positive --> 1
8. positive --> 1
9. positive --> 1
10. positive --> 1

Από όπου παρατηρούμε ότι και τα δέκα στοιχεία ανήκουν στα positive και όπως είναι λογικό ταξινομούνται όλα στο ίδιο label . Ωστοσο, δεν είναι δυνατόν από τα παραδείγματα αυτά, να διαπιστώσουμε την ύπαρξη της negative κλάσης που αντιστοιχίζεται στο label 0.

Εκτελώντας την ίδια διαδικασία και για dataset Semeval2017A, προκύπτουν τα ακόλουθα:

1. positive --> 2
2. negative --> 0
3. negative --> 0
4. negative --> 0
5. neutral --> 1
6. neutral --> 1
7. positive --> 2
8. negative --> 0
9. neutral --> 1
10. negative --> 0

Από αυτά, εύκολα διαπιστώνουμε ότι έχουμε τρείς κλάσεις όπως ήταν αναμενόμενο και αναλόγα την κλάση, καθένα αντιστοιχίζεται σε συγκεκριμένο label.

Βήμα 1.2: Λεκτική Ανάλυση (Tokenization)
ΖΗΤΟΥΜΕΝΟ 2

Έχωντας πλέον αντικατασησει τις κλάσεις με τα αντίστοιχα labels, δίνουμε στην συνάρτηση “SentenseDataset”, τα απαράιτητα δεδομένα, έτσι ώστε να γίνει η λεκτική ανάλυση των προτάσεων. Στην συνάρτηση αυτή, δίνονται ως παράμετροι οι προτάσεις του dataset, οι λίστες με τα labels για κάθε πρόταση, το dictionary που αντιστοιχέι τις λέξεις σε tokens και μια μεταβλητή AVG, που ορίζεται σαν υπερπαράμετρος και η χρησιμότητα της θα αναλυθεί στο επόμενο ερώτημα.

Με αυτόν τον τρόπο, δίνονται τα απαραίτητα στοιχέια που απαιτούνται ώστα να αρχικοποιηθούν στην “_init_” της “SentenceDataset” τα δεδομένα μας, αφού προηγηθεί μια προεπεξεργασία των προτάσεων μέσω της έτοιμης συνάρτησης “word_tokenize” της βιβλιοθήκης ntlk.

Βήμα 1.3: Κωδικοποίηση Παραδειγμάτων (Λέξεων)
ΖΗΤΟΥΜΕΝΟ 3

Για την ολοκλήρωση της κλάσης “SentenceDataset” απαιτέιται και η υλοποίηση της “_getitem_”. Σε αυτην την συνάρτηση γίνεται χρήση της υπερπαραμέτρου AVG, στην οποία έγινε αναφορά στο προηγούμενο ερώτημα και έχει περάσει πλέον στην συνατηση μας ως παράμετρος “self.avg”. Η παράμετρος αυτή καθορίζει το απόλυτο μήκος που θα επιλεγεί για την αναπαράσταση των προτάσεων μας, έτσι ώστε όλες οι προτάσεις να έχουν ακριβώς το ίδιο μήκος. Επομένως, η συνάρτηση αυτή αυτή υλοποιεί την αναπαράσταση των προτάσεων σε tokens και την μετατροπή τους σε διανύσματα συγκεκριμένου μήκους, ένω μαζί με αυτό το διάνυσμα, υπολογίζει και το πραγματικό μήκος της πρότασης (εφόσον αυτό είναι μικρότερο ή ίσο με AVG, αλλίως επιστρέφει AVG) αλλά και το label στο οποίο αντιστοιχίζεται. Αναφέρουμε ότι τα στοιχέια αυτά, έχουν υλοποιηθεί ως tensor, για να διευκόλυνση της διαδικασίας της επεξεργασίας τους στα ακόλουθα βήματα.

Επιλέγουμε AVG=20, και για τα πέντα πρώτα παραδείγματα του MR-train έχουμε την ακόλουθη απεικόνιση:

1. the rock is destined to be the 21st century's new " conan " and that he's going to make a splash even greater than arnold schwarzenegger , jean-claud van damme or steven segal . (tensor([1,1138,15,10454,5,31,1,5034,590,10,51,29,18513,29,6,13,19,10,223,5]), tensor(1), tensor(20))

2. the gorgeously elaborate continuation of " the lord of the rings " trilogy is so huge that a column of words cannot adequately describe co-writer/director peter jackson's expanded vision of j . r . r . tolkien's middle-earth .
   (tensor([1,78616,5135,10117,4,29,1,2371,4,1,6820,29,12305,15,101,1325,13,8,3236,4]), tensor(1), tensor(20))

3. effective but too-tepid biopic
   (tensor([2038,35,201535,34277,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]), tensor(1), tensor(4))

4. if you sometimes like to go to the movies to have fun , wasabi is a good place to start .
   (tensor([84,82,1072,118,5,243,5,1,2460,5,34,2906,2,66408,15,8,220,242,5,466]), tensor(1), tensor(20))

5. emerges as something rare , an issue movie that's so honest and keenly observed that it doesn't feel like one .
   (tensor([12398,20,646,2349,2,30,496,1006,13,10,101,6082,6,23499,4583,13,21,261,71,999]), tensor(1), tensor(20))

Βήμα 2.1: Embedding Layer
ΖΗΤΟΥΜΕΝΟ 4

Για την δημιουργία του νευρωνικού δικτύου, απαιτείται η δημιουργία του embedding layer, όπου βισκονται οι είσοδοι που θα χρησιμοποιηθούν κατα την εκπάιδευση του νευρωνικού. Για τον σκοπό αυτό, αρχικοποιούμε στην “_init_” τα βάρη από τα προεκπαιδευμένα embeddings και καθορίζουμε να παγώσει το embedding layer, δηλαδή τα βάρη του δικτύου να μην ενημερωθούν με τις εντολές

1. “self.trainable_emb=trainable_emb, όπου έχει αρχικοποιηθεί “ trainable_emb=False”.
2. self.embedding.weight.requires_grad=self.trainable_emb”

Τα προεκπαιδευμένα word embeddings, αποτελούν διανύσματα τα οποία έχουν εξαχθεί από εκπαίδευση νευρωνικών από κείμενα πολλών corpus. Έτσι αρχικοποιώντας το embedding layer με αυτά, εξοικονομούμε σημαντικό χρόνο αλλα και επιτυγχάνουμε μεγαλύτερη ακρίβεια αναπαράστασης.

Η προεκπαίδευση τους, μας επιτρέπει να μην χρείαζεται να ξεκινήσουμε από τυχαία διανύσματα και να επαναπροσδιορίζουμε κατα την εκπαίδευση τα βάρη τους. Γι’ αυτόν τον λόγο, είναι απαραίτητο να παγώσουμε το embedding layer κατα την εκπαίδευση, διότι από την μία δεν χρειάζεται επιπλέον εκπάιδευση (είναι ήδη προεκπαιδευμένα) και από την άλλη τα αρχικά embeddings επιθυμείτε να είναι σταθερα και να μην μεταβάλλεται από εποχή σε εποχή, καθώς θεωρούνται σταθερό σημείο αναφοράς.

Βήμα 2.2: Output Layer(s)
ΖΗΤΟΥΜΕΝΟ 5

Στο μέρος αυτό, ορίζουμε τα διαφορετικά μέρη που θα αποτελέσουν το νευρωνικό μας δίκτυο. Για τον σκοπό αυτό, ορίζουμε το μέγεθος του hidden layer, όπου τίθεται ως υπερπαραμετρος (DNN_HIDDEN_LAYER), το Dropout, την μη γραμμική συνάρτηση ενεργοποίησης, όπου στην περίπτωσή μας έχει επιλεγεί η ReLU, το linear layer, όπου κάνει την απεικόνιση τις εισόδου στο hidden layer και ,τέλος, το στάδιο εξόδου (που είναι και αυτό linear) που κάνει την τελική απεικόνιση πό το hidden layer στις κλάσσεις που υπάρχουν, των οποίων το πλήθος έχει δωθεί σας παραετρος στο μοντελο.

Η μη γραμμική συνάρτηση ενεργοποίησης έχει ώς στόχο την ομαλοποίηση των δεδομένων πριν περάσουν από το τελικό γραμμικό στάδιο (output layer), ούτως ώστε να μην υπάρχει μεγάλη διασπορά στις τιμές που αυτά καταλαμβάνουν. Αναφέρεται ότι υπάρχουν πολλές μη γραμμικές συναρτήσεις ενεργοποιήσης με πιο διαδεδομένες την ReLU = max(0, xi) και την Tanh. Η χρήση των γραμμικών μετασχηματισμών σχετίζεται με την δημιουργία νευρώνων στο δίκτυο μας. Με την προσθήκη, επιπλέον γραμμικών επιπέδων , αυξάνονται οι συνολικοί νευρώνες που υπάρχουν στο δίκτυο μας και παίρνουν μέρος στην εκπαίδευση του μοντέλου μας. Περισσότεροι νευρώνες, συνεπάγεται περισσότεροι παράμετροι (στα κρυφά layers) που παίρνουν μέρος για την εξαγωγή του τελικού αποτελέσματος. Με αυτόν τον τρόπο, επιτυγχάνεται πιο αποτελεσματική εκπαίδευση και το τελικό μας μοντέλο είναι πιο αποδοτικό.

Βήμα 2.3: Forward pass
ΖΗΤΟΥΜΕΝΟ 6
Στο ερώτημα αυτό, γίνεται κατασκευή της διαδικασίας “forward” του νευρωνικού μας δικτύου. Για αυτόν τον σκοπό, αρχικοποιούμε τα embeddings μας, ύστερα, υπολογίζουμε το mean pooling των προτάσεων με τον τρόπο που περιγράφεται και δημιουργούμε τις αναπαραστάσεις μας. Τις αναπαραστάσεις αυτές τις περνάμε σειριακά από linear layer, εφαρμόζουμε την Dropout και μετέπειτα την ReLU, και τέλος, αφού τα περάσουμε από το outpur layer, λαμβάνουμε τις προβεψεις μας.

Η χρήση του mean pooling για την απεικόνικη των προτάσεων υποδηλώνει ότι η κάθε πρόταση θα είναι ο μέσος όρος των λέξεων που την αποτελούν. Έτσι, για κάθε πρόταση λαμβάνεται ένα διάνυσμα που προκύπτεί από το μέσο όρο των διανυσμάτων των λέξεων που την αποτελούν. Η απεικόνιση αυτή έχει και γλωσσολογική υπόσταση. Αν σκεφτεί κανείς, τα δομικά στοιχεία κάθε πρότασης είναι οι λέξεις που την αποτελούν. Έτσι λαμβάνοντας μια ενιαία αναπαράσταση αυτών, μπορούμε να πούμε ότι γίνεται αντιπροσωπευτική απεικόνιση της πρότασης.

Ωστόσο, μπορούμε να διαπιστώσουμε ότι η παραπάνω απεικόνιση του mean pooling μπορεί, εν γένει, να πάσχει από αδυναμίες. Σε πολλές περιπτώσεις το κύριο συναίσθημα που προσδίδει κάθε πρόταση δεν βρίσκεται στο σύνολο των λέξεων αλλά σε ορισμένες μόνο λέξεις που επικαλύπτουν τις υπόλοιπες. Έτσι, θα μπορούσαμε να εφαρμόσουμε διαφορετικούς τρόπους αναπαράστασης, ούτως ώστε να έχουμε πιο αντιπροσωπευτική απεικόνιση, όπως θα δούμε στο κύριως μέρος της άσκησης.

Βήμα 3.1: Φόρτωση Παραδειγμάτων (DataLoaders)
ΖΗΤΟΥΜΕΝΟ 7
Στο ερώτημα αυτό φορτώνουμε τα παραδείγματα στην κατάλληλη μορφή με χρήση της κλάσης “Dataloader” της βιβλιοθήκης torch.utils.data. Στην συνάρτηση αυτή πέρα από το set που δίνουμε, περνάμε μία υπερπαράμετρο με το μέγεθος του batch (BATCH_SIZE) και προσδιορίζουμε να γίνεται suffle των mini-batches στα δεδομένα εκπαίδευσης κάθε εποχή.

Σημαντικό ρόλο στην εκπαίδευση του νευρωνικού έχει η επιλογή της υπερπαραμέτρου BATCH_SIZE, που ορίζει το μέγεθος των mini-batches. Έχει αποδειχθεί ότι, τα μικρό batch_size δίνει καλύτερα αποτελέσματα και η σύγκλιση γίνεται πιο γρήγορα. Ωστόσο, δεν είναι εγγυημένη η σύγκλιση στο global optima. Από την άλλη μεριά, επιλέγοντας μεγάλο μέγεθος για τα batches, κινδυνεύουμε να πάθουμε overfiting και να μην έχουμε αποδοτικό νευρωνικό δίκτυο σε άγνωστα δεδομένα.

Αλλό ένα ουσιαστικό χαρακτηριστικό που αποτρέπει το overfitting είναι το suffle. Επιλέγοντας σε κάθε εποχή να ανακατεύεται η σειρά των mini-batches, καταφέρνουμε να μην εκπαιδεύουμε το νευρωνικό μας σε δεδομένα με την ίδια σειρά, γεγονός που οδηγεί στην απόδοση του μοντέλου και την γενίκευση σε άγνωστα δεδομένα.

Βήμα 3.2: Βελτιστοποίηση
ΖΗΤΟΥΜΕΝΟ 8
Στο μέρος αυτό όπως υποδηλώνει και η εκφώνηση, αρχικά ορίζουμε το μοντέλο μας με χρήση της κλάσης “BaselineDNN” που κατασκευάστηκε στα προηγούμενα ερωτήματα. Στην συνέχεια, επιλέγεται ως κριτήριο για αμφότερα τα μοντέλα επιλέγεται το CrossEntropyLoss, καθώς και στο 3-class-classifier και στο binay-classifier στο τελευταίο layer προβάλουμε τα δεδομένα στο χώρο των κλάσεων. Επομένως, δεν συνίσταται η χρήση του BCEWithLogitsLoss. Επίσης, επιλέγονται οι παράμετροι, οι οποίοι είναι trainable, με την εντολή:

- “parameters = [parameter for parameter in list(model.parameters()) if parameter.requires_grad==True]”

Τέλος, επιλέγεται ως αλγόριθμος βελτιστοποίησης ο Adam. Έτσι έχουμε ορίσει όλα τα απαιτούμενα μέρη, που χρειάζονται για να οδηγηθούμε στην εκπαίδευση του νευρωνικού.

Βήμα 3.3: Εκπαίδευση
ΖΗΤΟΥΜΕΝΟ 9
Για το μέρος αυτό, γίνεται η υλοποίηση της εκπαίδευσης του νευρωνικού. Για αυτόν τον σκοπό, ορίζουμε δύο διαφορετικές συναρτήσεις, μία για το train_set και μια για το dev_set. Στην συνάρτηση, για τα δεδομένα της εκπαίδευσης, αφού θεσουμε το δίκτυο σε “mode=train” και αφού αρχικοποιούμε τις παραγώγους στο μηδεν και υπολογίζοντας για κάθε batch τις εξόδους, το σφάλμα και εν συνεχεία, τις παραγώγους, ανανεώνουμε τα βάρη και υπολοιούμε την ίδια διαδικάσία για ολά τα batches για κάθε εποχή.

Στην συνάρτηση, για τα δεδομένα αξιολόγησης, η μόνη διαφορά που εντοπίζεται είναι ότι, η διαδικασία, που περιγράφτηκε γίνεται μέσα στην εμβέλεια της εντολής:

- “with torch.no_grad()”,

όπου δεν αφήνει την ανανέωση των βαρών, αφού δεν υπολογίζονται οι παράγωγοι, γεγονός αναμενόμενο αφού στο evaluation set, απαγορεύεται ο επαναπροσδιορισμός των βαρών.

Βήμα 3.4: Αξιολόγηση
ΖΗΤΟΥΜΕΝΟ 10
Σε συνδυασμό με όλα τα παραπάνω για την αποφυγή του overfiting του μοντέλου μας, έχει υλοποιηθεί η συνάρτηση “EarlyStopping” , όπου δείχνει ανοχή σε συγκεκριμένο πλήθος εποχών(ίσο με την παράμετρο PATIENCE) δίχως σημαντική πτώση στο loss. Αν δε δείξει βελτίωση, η εκπαίδευση σταματά και αποθηκευμένο το καλύτερο μοντέλο με κριτήριο το loss.

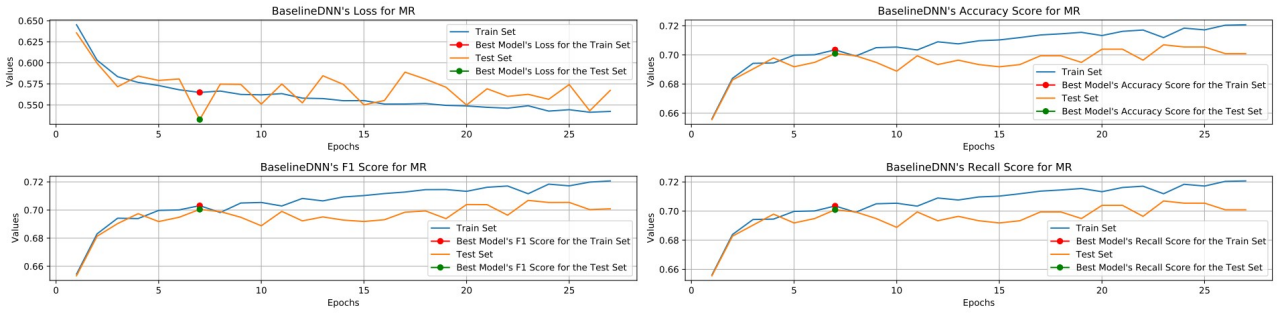
Παρακάτω ακολουθή λίστα με τις υπερ-παραμέτρους που χρησιμοποιήθηκαν:

EMB_DIM = 50
AVG = 50
DNN_HIDDEN_LAYER = 50
PATIENCE = 20
EMB_TRAINABLE = False
BATCH_SIZE = 128
EPOCHS = 50
ETA = 1e-3

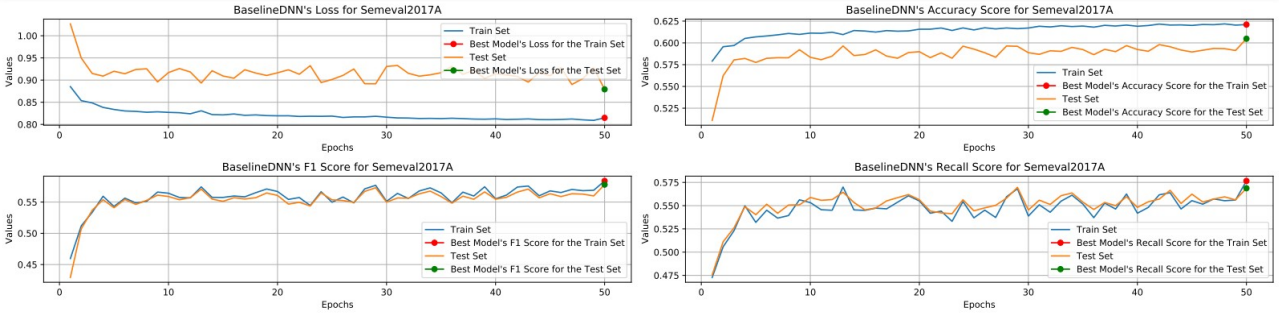
και με χρήση των embeddings: "glove.6B.50d.txt"

Για τις υπερπαραμέτρους αυτές προκύπτουν τα ακόλουθα διαγράμματα:

- Για το dataset “MR”:



- Για το dataset “Semeval2017A”:



<u>Σχόλια:</u>

1. Παρατηρούμε ότι, και για τα δύο datasets έχουμε πτώση του loss με το πέρασμα του χρόνου, όσο αφορά το train set.
2. Για το test set, μπορεί να παρατηρήσει κανείς ότι, για το “MovieReview”, το loss κυμαίνεται περίπου στην ίδια τιμή, ενώ για το dataset με τα Tweets, έχουμε μια ένδειξη overfitting, καθώς το loss δείχνει να αυξάνεται ελαφρώς από την 35η εποχή και μετά.
3. Όσο, αφορά τις μετρικές, όπως αναμενόταν το train set δείχνει ανοδική πορεία, όπως και το test set με μέγιστη τιμή 0.7 και 0.6(accuracy_score) για τα “MovieReview”, Tweets αντίστοιχα.
4. Προφανώς, η επιλογή embeddings με μεγαλύτερες διαστάσεις, θα πρόσδιδαν περισσότερη πληροφρία και καλύτερη απόδοση στο μοντέλο μας.

ΚΥΡΙΩΣ ΜΕΡΟΣ

Με την ολοκλήρωση της προπαρασκευής, έχοντας κατασκευάσει και εκπαιδεύσει το απλό νευρωνικό δίκτυο, είμαστε σε θέση να συνεχίσουμε με την κατασκευή πιο σύνθετων νευρωνικών δικτύων. Έχοντας ως βάση το νευρωνικό δίκτυο της προπαρασκευής, κάνουμε σε κάθε περίπτωση τις απαραίτητες αλλαγές και διαφοροποιήσεις ούτως ώστε να κατασκευάσουμε κάθε φορά το ζητούμενο νευρωνικό δίκτυο.
Κάθε νέο νευρωνικό δίκτυο εντάσσεται ως μία νέα κλάση στο αρχείο “models.py”, όπου εκέι περιγράφεται η αρχιτεκτονική του. Τα υπόλοιπα scripts της εκπαίδευσης και του πρόωρου τερματισμού παραμένουν ίδια καθώς έχουν δημιουργηθεί με τέτοιον τρόπο ώστε να εκτελούν την διαδικασία τους αμετάβλητα και ανεξάρτητα από το εκάστοτε μοντέλο. Έτσι μπορούμε να αναλύσουμε για κάθε ένα από τα δημιουργηθέντα μοντέλα την λειτουργία και τα αποτελέσματα που λαμβάνουμε από αυτά.

Για τα διαγράμματα που προκύπτουν στα υπόλοιπα υποερωτηματα, έχουν επιλεγεί οι εξής υπερπαράμετροι που αναφέρφηκαν και στην “Προπαρασκευή” με την μόνο διαφορά ότι προσιθενται τα ακόλουθα:

- BoW = “”/”BoW”, ανάλογα με το αν επιθημούμε να ενσωματώσουμε στο μοντέλο μας χαρακτηριστικα BoW(δίνουμε τη λέξη “BoW” για True και “κενη συμβολοσειρά” για False για τεχνικούς λόγους στον κώδικα
- LSTM_HIDDEN_LAYER = EMB_DIM, για τα LSTM νευρωνικά
- BI_HIDDEN_LAYER = EMB_DIM*2, για τα αμφίδρομα νευρωνικά

και γίνεται χρήση των embeddings: "glove.6B.50d.txt".

Τέλος, οι αξιολογήσεις γίνονται στο dataset MR.

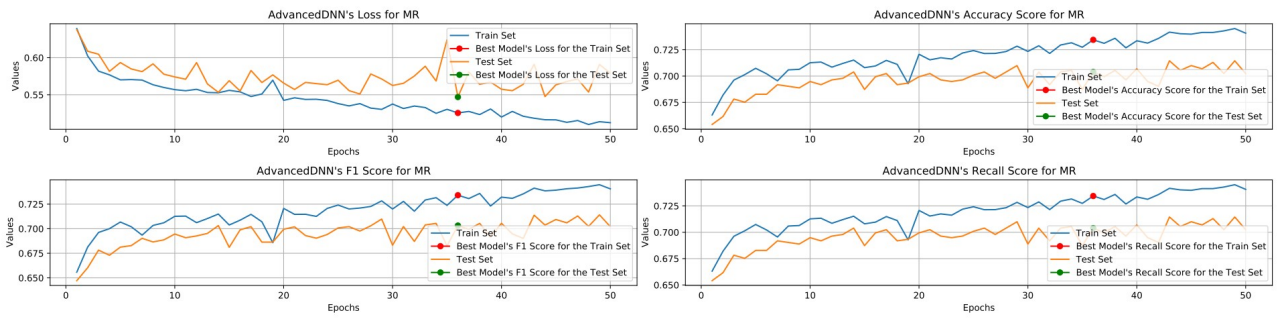
ΕΡΩΤΗΜΑ 1.1

Στο ζητούμενο αυτό, ζητείται να μεταβληθεί η αναπαράσταση κάθε πρότασης που χρησιμοποιείται. Έτσι στο παρόν μοντέλο, ως αναπαράσταση κάθε πρότασης u επιλέγεται η συνένωση (concatenation) του μέσου όρου (mean pooling) και του μεγίστου ανά διάσταση (max pooling) των word embeddings κάθε πρότασης, E = (e1, e2, ..., eN).

Η μαθηματική περιγραφή της παραπάνω έκφρασης είναι η εξής: u = [mean(E)||max(E)].

Η κλάση που περιγράφει το μοντέλο αυτό είναι η “AdvancedDNN”.

Έτσι χρησιμοποιώντας την ίδια δομή και την ίδια εκπαίδευση που έχουμε χρησιμοποιήσει και στην προηγούμενη περίπτωση λαμβάνουμε τα ακόλουθα αποτελέσματα για το επιλεγμένο dataset “MR”:



Από τα αποτελέσματα που λαμβάνουμε, διαπιστώνουμε ότι το νέο μοντέλο είναι πιο αποδοτικό σε σχέση με το αρχικό, γεγονός που επηρεάζεται από την νέα αναπαράσταση των προτάσεων που χρησιμοποιούμε. Ωστόσο, η βελτίωση στην ακρίβεια που επιτυγχάνεται, είναι εις βάρος του χρόνου εκπαίδευσης, όπου απαιτείται ελάχιστα παραπάνω χρόνος για το “AdvancedDNN” σε σχέση με το αρχικό.

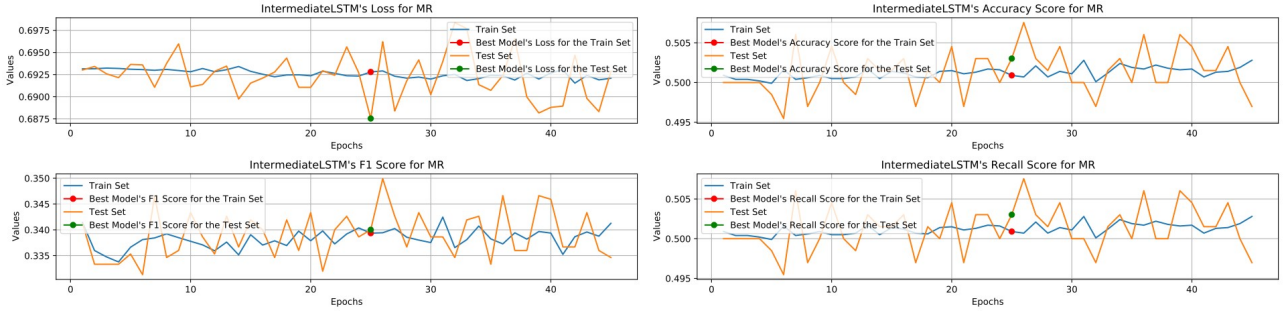
ΕΡΩΤΗΜΑ 1.2

Με την αναπαράσταση mean-pooling, μπορούμε να διαπιστώσουμε ότι η κάθε πρόταση αναπαρίσταται ως ο μέσος όρος των λέξεων που την αποτελούν. Με τον τρόπο αυτό, λαμβάνουμε μια αντιπροσωπευτική απεικόνιση της πρότασης στον χώρο. Συμπεριλαμβάνοντας μαζί με το mean-pooling και το max-pooling, ενσωματώνουμε στην προηγούμενη “μέση” αναπαράσταση της πρότασης την επίδραση σημαντικών λέξεων που την αποτελούν. Έτσι λαμβάνουμε ένα διάνυσμα, το οποίο έχει μεν το διπλάσιο μήκος σε σχέση με το αρχικό αλλά είναι πιο αντιπροσωπευτικό. Αυτό συμβαίνει, διότι, η τελική απεικόνιση της κάθε πρότασης περιέχει πληροφορία από όλες τις λέξεις που την αποτελούν, φυσικό ακόλουθο αφού οι λέξεις αποτελούν τα δομικά συστατικά της κάθε πρότασης, αλλά και από λέξεις που έχουν μεγαλύτερο συναισθηματικό περιέχομενο και υπερκαλύπτουν τις υπόλοιπες, λαμβάνοντας έτσι μια πιο αποδοτική απείκόνιση τους. Η παραπάνω διαπίστωση είναι εμφανής και από το γεγονός ότι τα αποτελέσματα που λαμβάνουμε για τις επιδόσεις του μοντέλου είναι βελτιωμένες σε σχέση με το αρχικό.

ΕΡΩΤΗΜΑ 2.1

Στο ερώτημα αυτό κατασκευάζουμε ένα μοντέλο LSTM. Είναι γνωστό ότι τα LSTM μοντέλα αποτελούν βελτίωση των RNN, διότι σε αντίθεση με τα δεύτερα αντιμετωπίζουν αποτελεσματικά το πρόβλημα των vanishing/exploding gradients. Σε πρώτη φάση, κατασκευάζουμε το LSTM χρησιμοποιώντας ως αναπαραστάσεις την τελευταία έξοδο του LSTM h_N ως την αναπαράσταση του κειμένου u.

Για την κατασκευή του νευρωνικού καλούμε την κλάση “nn.LSTM” της βιβλιοθήκης “pytorch”. Έτσι δημιουργούμε την κλάση “IntermediateLSTM” και ακολούθως, παραθέτουμε τα αποτελέσματα που προκύπτουν για το dataset “MR”:

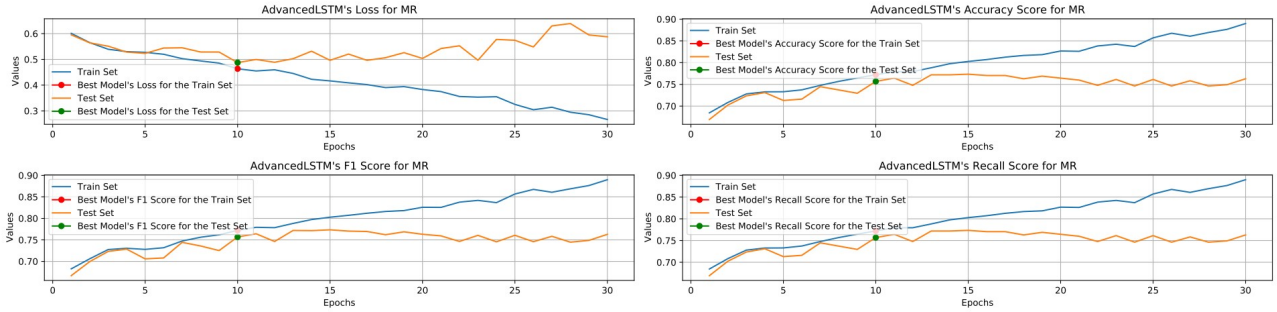


ΕΡΩΤΗΜΑ 2.2

Επεκτείνουμε την προηγούμενη ανάλυση, και δημιουργούμε μία νέα αναπαράσταση για τις προτάσεις που χρησιμοποιούμε ως εισόδους. Συγκεκριμένα, σε αυτό το μοντέλο ως αναπαράσταση του κειμένου u λαμβάνουμε την συνένωση της τελευταίας εξόδου του LSTM h_N , του μέσου όρου των εξόδων του LSTM, και του μεγίστου ανά διάσταση των εξόδων του LSTM.

Η μαθηματική περιγραφή της παραπάνω έκφρασης είναι: u = [hN ||mean(E)||max(E)]

Το παρόν μοντέλο κωδικοποιείται στην κλάση “AdvancedLSTM”, για την οποία λαμβάνουμε τα ακόλουθα για το dataset “MR”:



Από τα παραπάνω, διαπιστώνουμε ότι το μοντέλο μας παρουσιάζει σημαντική βελτίωση. Ωστόσο, όπως και πριν η πλήρης εκπαίδευση του, μέχρι τον τερματισμό όλων των εποχών διαρκεί περισσότερο χρόνο. Μια, αποδοτική λύση του προβλήματος αυτού είναι ο μηχανισμός early stopping, ούτως ώστε να σταματήσει η εκπαίδευση του μοντέλου στο σημείο που θα αρχίσει να γίνεται overfiting.

Παρ’ολο το EarlyStoppig(Reguralization) μπορεί να παρατηρήσει κανείς, το συγκεκριμένο γεγονός. Εξάλλου, τα LSTM τείνουν να παθαίνουν overfitting λόγω της μεγάλης ροής πληροφορίας, την οποία διαχειρίζονται για το εκάστοτε train dataset.

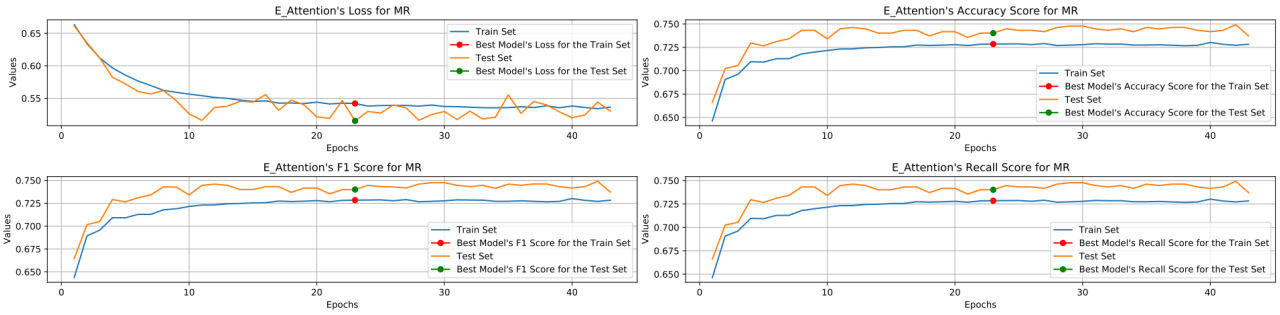
<u>ΕΡΩΤΗΜΑ 3.1</u>

Στο ερώτημα αυτό, ζητείται η μοντελοποίηση του μηχανισμού attention. Με την χρήση του attention, δίνεται περισσότερη ή λιγότερη σημασία σε ορισμένες εισόδους / επίπεδα του νευρωνικού. Η μέθοδος αυτή αποτελεί και τρόπο αντιμετώπισης των vanishing/exploding gradients.

Αρχικά γίνεται κατασκευή του νευρωνικού που περιγράφεται από τις σχέσεις:

$$\upsilon_i = \tanh\left(W e_i + b\right) \qquad \alpha_i = \frac{\exp\left(\upsilon_i\right)}{\sum_{t=1}^{N} \exp\left(\upsilon_t\right)} \qquad u = \sum_{i=1}^{N} \alpha_i e_i$$

Κατασκευάζουμε την κλάση “E_Attention”, ενσωματώνοντας την υλοποίηση των attention που μας δίνεται και έτσι κατασκευάζουμε το ζητούμενο νευρωνικό δίκτυο που δίνει τα παρακάτω αποτελέσματα για το dataset “MR”:



Σχόλια:

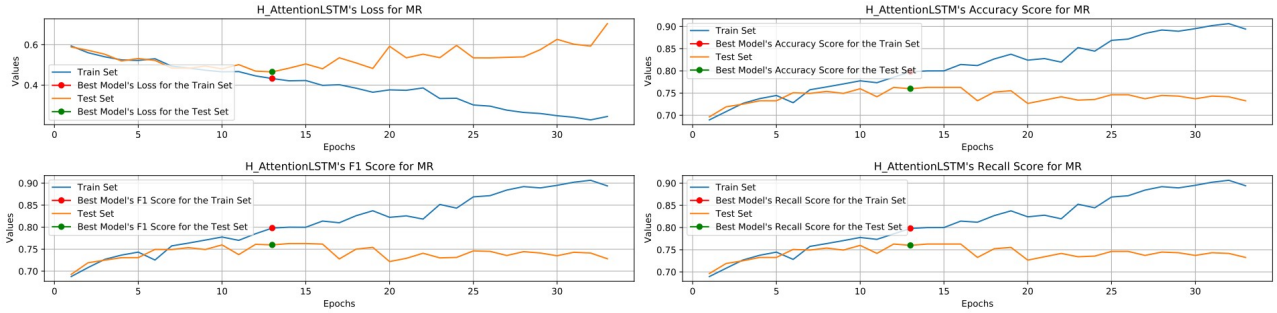
1. Πολύ καλύτερη επίδοση από πλευράς accuracy, αλλα ταυτοχρόνως σημαντική πτώση πτώση του loss. Επομέωνως, θα μπορούσε να πει κανείς ότι, η τεχνική Attention αντιμετωπίζει το πρόβλημα του overfitting στο συγκεκριμένο παράδειγμα.

ΕΡΩΤΗΜΑ 3.2

Για την κατασκευή του μοντέλου αυτού ακολουθούμε την ίδια διαδικασία με αυτήν που αναλύθηκα παραπάνω, με την μόνη διαφορά ότι εδώ το μοντέλο μας περιγράφεται από τις εξής εκφράσεις:

$$\upsilon_i = \tanh\left(W h_i + b\right) \qquad \alpha_i = \frac{\exp\left(\upsilon_i\right)}{\sum_{t=1}^{N} \exp\left(\upsilon_t\right)} \qquad u = \sum_{i=1}^{N} \alpha_i h_i$$

Όπως και πριν, κατασκευάζουμε την κλάση “H_AttentionLSTM”, ενσωματώνοντας την υλοποίηση των attention που μας δίνεται και έτσι δημιουργούμε το ζητούμενο νευρωνικό δίκτυο που δίνει τα παρακάτω αποτελέσματα για το dataset “MR”:

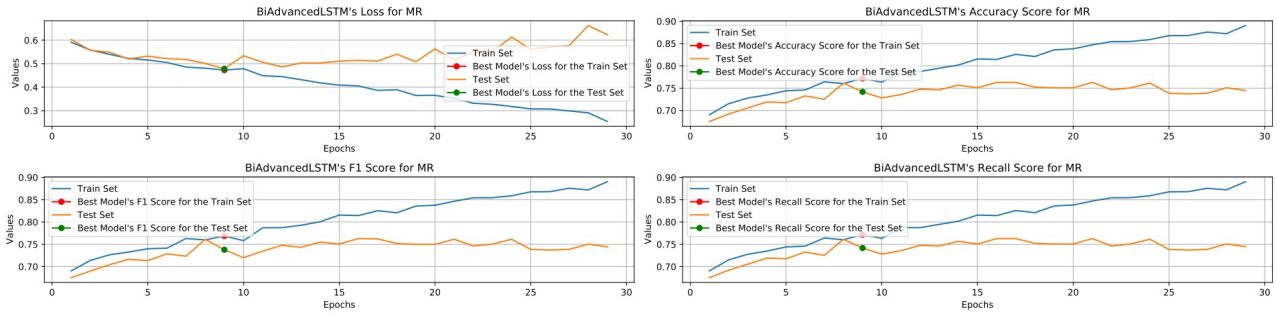


Από τα αποτελέσματα που προκύπτουν διαπιστώνουμε μείωση του loss και μια μικρή βελτίωση της απόδοσης για το καλύτερο μοντέλο μας, σε σχέση με τις προηγούμενες υλοποιήσεις. Το γεγονός αυτό είναι αναμενόμενο, αφού έχουμε ενσωματώσει σε αυτό νέα χαρακτηριστικά ούτως ώστε να γίνεται καλύτερη εκπαίδευση. Δυστυχώς, όπως και στα υπόλοιπα LSTM, προκύπτει δυσκολία στη γενίκευση.

ΕΡΩΤΗΜΑ 4.1

Σε αυτό το ερώτημα, ζητείται να συνδυάσουμε τα αμφίδρομα LSTMs με τον μηχανισμό attention.
Με τα αμφίδρομα δίκτυα μπορούμε να συνδυάσουμε τόσο την backward όσο και την forward πληροφορία, έχοντας μία πιο αποτελεσματική εκπαίδευση. Ενώ σε συνδυασμό με τον μηχανισμό attention δίνεται περισσότερη βάση στα στάδια όπου κρίνεται απαραίτητο. Συνεπώς τα αποτελέσματα που λαμβάνουμε θα πρέπει να παρουσιάζουν βελτίωση σε σχέση με τα υπόλοιπα μοντέλα.

Για το πρώτο μοντέλο δημιουργούμε την κλάση “BiAdvancedLSTM” και προκύπτουν τα ακόλουθα αποτελέσματα για το dataset “MR”:

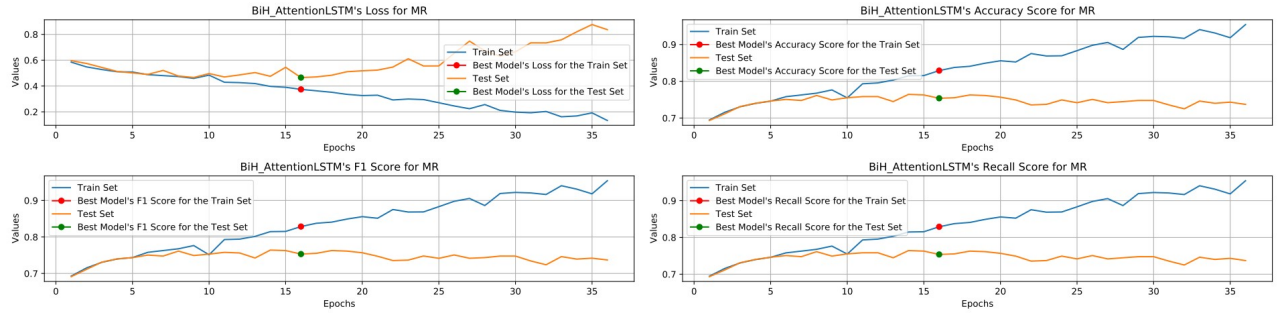


Σχόλια:

1. Όμοιες παρατηρήσει με τα υπόλοιπα LSTM. Το Attention layer δεν δείχνει να μπορεί να αντιμετωπίσει αυτό το πρόβλημα, τουλάχιστον με την απλή υλοποίηση, που πραγματοποιήσαμε.

ΕΡΩΤΗΜΑ 4.2

Για το δεύτερο μοντέλο δημιουργούμε την κλάση “BiH_AttentionLSTM” και προκύπτουν τα ακόλουθα αποτελέσματα για το dataset “MR”:



Από τα αποτελέσματα, διαπιστώνουμε ότι όπως ήταν αναμενόμενο τα αμφίδρομα μοντέλα που προκύπουν έχουν καλύτερα χαρακτηριστικά(το καλύτερο μοντέλο αυτής της κλάσης δίνει το μικρότερο loss) σε σχέση με τα αντίστοιχα μονόδρομα μοντέλα, και επιτυγχάνουν μικρότερο loss και μεγαλύτερη ακρίβεια.

Για να καταλήξουμε σαυτο το συμπέρασμα, δεν αρκεστήλαμε στην οπτικοποίηση, που προσδίδουν τα διαγράμμα, αλλά ορίσαε την μεταβλητή:

- BEST = {“MR”: []}

Η χρήση της φαίνεται στο παρακάτω κομμάτι κώδκα.

```
best = early.get_best()
if BEST[DATASET] == []:
    BEST[DATASET] = [MODEL, best.copy(), copy.deepcopy(model)]

elif BEST[DATASET][1]['loss'][1] > best['loss'][1]:
    BEST[DATASET] = [MODEL, best.copy(), copy.deepcopy(model)]
```

Το παραπάνω, έχει τοποθετηθεί εντός του for loop της εκπαίδευσης-αξιολόγησης και με βάση το loss του προηγούμενου καλύτερου μοντέλου(μεταξύ όλων των διαφορετικών ειδών που αναπτύξαμε) συγκρίνει αν έχει υπάρξει βελτίωση ή όχι. Αποθηκεύει το όνομα, τα χαρακτηριστικά(εποχή, loss, accuracy,κλπ) και ένα αντίγραφο του μοντέλου.

ΕΡΩΤΗΜΑ 5.1

Η εύρεση του καλύτερου μοντέλου περιγράφηκε στο προηγούμενο ερώτημα. Το αρχείο με τις προβλέψεις του καλύτερο δικτύου είναι το αρχείο “predictions.txt”.

ΕΡΩΤΗΜΑ 5.2

Γιαν την οπτικοποιήση των βαρών των κατανομών του attention μέσω του NeAt-vision είναι απαραίτητη η δημιουργία των “.json” αρχείων. Για τον σκοπό αυτό, δημιουργείται το script “createdata.py”, όπου εκτελεί αυτήν την διαδικασία. Έτσι δημιουργούμε τα απαράιτητα αρχεία για τα μοντέλα στα οποία ενσωματώνουμε τους μηχανισμκούς attention και στην συνέχεια μέσω του NeAt-vision προκύπτουν τα ακόλουθα (στις παρακάτω φωτογραφίες εμφανίζονται μόνο ορισμένες τυχαίες προτάσεις, σαν παραδείγματα) :

- Για το “E_Attention”:

Παρατηρώντας συνολικά τα αποτελέσματα που προκύπτουν, διαπιστώνουμε σε αυτό το μοντέλο τα μεγαλύτερα βάρη δίνονται σε λέξεις που είναι συνθαισθηματικά φορτισμένες και η συναισθηματική ανάλυση γίνεται δίνοντας μεγαλύτερη προσοχή σε αυτές. Αυτό οδηγεί όμως , πολλές φορές σε λανθασμένα συμπεράσματα, όπως διαπιστώνουμε αν συγκρίνουμε τις προβλέψεις με τα labels.

- Για το “H_AttentionLSTM”:

Από τα παραπάνω διαπιστώνουμε ότι στη περίπτωση του “H_AttentionLSTM” η κατανομή των βαρών είναι πιο διασκορπισμένη σε όλες τις λέξεις της πρότασης με αποτέλεσμα να δίνεται πιο “ορθα” η σημαντικότητα της κάθε λέξης με αποτέλεσμα να οδηγούμαστε σε ορθότερες προβλέψεις περισσότερες φορές σε σχέση με το προηγούμενο μοντέλο.

- Για το “BiH_AttentionLSTM”:

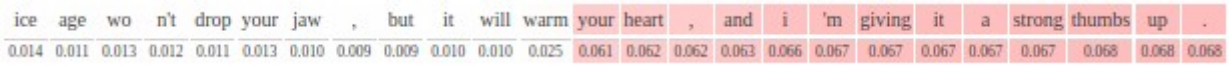
Τέλος παρατίθενται και τα ποτελέσματα που προκύπτουν για το “BiH_AttentionLSTM”, έτσι ώστε να επισημανθεί ότι ο συνδυασμός των αμφίδρομων LSTMs με τον μηχανισμό του attention δίνει την καλήτερη κατανομή λαθός, επιτυγχάνοντας παράλληλα την καλύτερη προβλεπτική ικανότητα σε σχέση με όλα τα προηγούμενα μοντέλα που αναλύσαμε.

ΕΡΩΤΗΜΑ 5.3

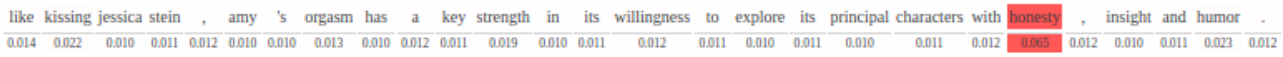
Από τα αποτελέσματα που προκύπτουν από το NeAt-vision μπορούμε να διαπιστώσουμε διαφορα χαρακτηριστικά για τα μοντέλα που χρησιμοποιούν τον μηχανισμό του attention και για την κατανομή των βαρών που δίνεται σε κάθε λέξη. Μερικά από αυτά είναι:

1. Παρατηρέιται ότι σε αμφότερα τα μοντέλα “E_Attention” και “H_AttentionLSTM” δίνεται μεγαλύτερο βάρων στις λέξεις που έχουν συναισθηματικό περιέχομενο (όπως honestly, seeling, wram, old κ.α.) με την μόνη διαφορά ότι στο “H_AttentionLSTM” η κατανομή αυτή γίνεται με πίο αποτελεσματικό τρόπο, έτσι ώστε να αοδηγηθούμε σε ορθότερες προβλέψεις.

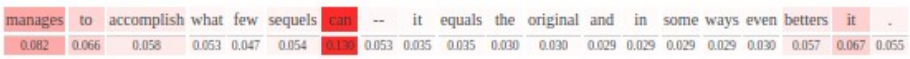
2. Σε κάποιες προτάσεις παρατηρείται ότι μέρη της πρότασης που δεν έχουν κάποια ουσιστική αξία στην συναισθηματική ανάλυση της, λαμβάνουν βάρη μεγαλύτερα από άλλες ουσιαστικές λέξεις. Χαρακτηριστικό παράδειγμα αποτελούν οι προασεις στις οποίες τα αρθρα και τα σημεία στήξης έχουν μεγαλύτερο συντελεστή σε σχέση με τις υπόλοιπες λέξεις. Το πρόβλημα αυτό, όπως θα δούμε και παρακάτω, αντιμετοπίζεται με την στάθμιση των λέξεων μέσω των συντελεστών tfi-df.



3. Τέλος, παρατηρείται ότι στις περισσότερες περιπτώσεις το αποτέλεσμα της προβλεπης σχετίζεται άμεσα με το πως έχουν κατανεμηθεί τα βάρη στις λέξεις. Στις περισσότερες προτάσεις, όπου οι λέξεις που περιέχουν συναισθήματα και αποτελούν την βάση της πρότασης πέρνουν μεγάλο βάρος, τοτε οι προβλέψεις τείνουν να έιναι ορθές. Σε αντίθεση, όταν τα μεγαλύτερα βάρη τοποθετούνται σε λέξεις που δεν έχουν κάποια ουσιατική θέση στην πρόταση, τότε για αυτές τις προτάσεις λαμβάνουμε αρκετές λανθασμένες προβλέψεις.



Label	**1 (😍)**
Prediction	**1 (😍)**



Label	**1 (😍)**
Prediction	**0 (😡)**

ΕΡΩΤΗΜΑ 6.1

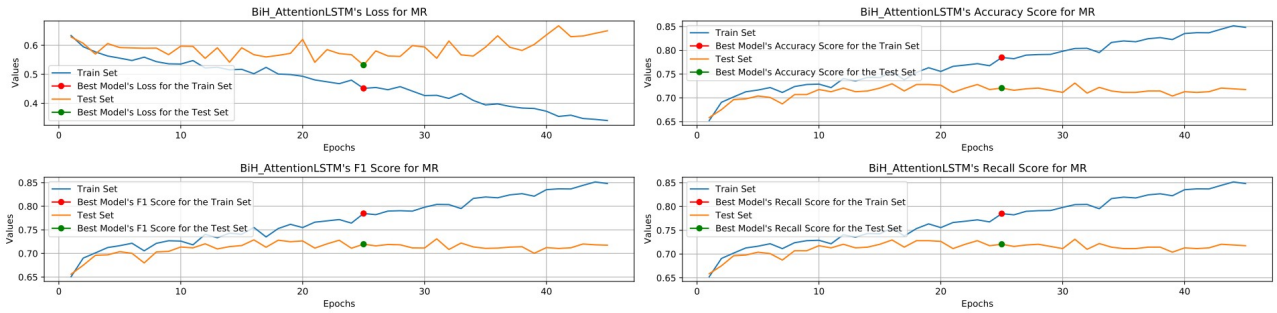
Στο ερώτημα αυτό γίνεται μία προσπάθεια να συνδυαστούν τα χαρακτηριστικά που εξάγει το νευρωνικό δίκτυο (αναπαράσταση) με τα χαρακτηριστικά tfi-df που μπορούν να εξαχθούν για τις λέξεις. Για να συμβεί το παρόν εξάγουμε από το διάνυσμα του X_train, τα χαρακτηριστικά tfidf με χρήση των συναρτήσεων “CountVectorizer” και “TfidfTransformer” της βιβλιοθήκης “scikit-learn”.

Επισημαίνεται ότι ο όρος tf (Term Frequencies) αναφέρεται στην συχνότητα εμφάνισης των λέξεων στο κείμενο μας, ενώ ο όρος “idf” (Inverse Document Frequency) που χρησιμοποιείται για την αντιστάθμιση του tf και υπολογίζει τις λέξεις που εμφανίζονται σε πολλά κείμενα του corpus μας. Έτσι εξάγονται οι μετρικές tfidf.

Για να είναι δυνατόν να εφαρμόσουμε την μέθοδο BoW σε όλα τα μοντέλα των νευρωνικών που έχουμε υλοποιήσει, επιλέγουμε να δημιουργήσουμε μία υπερπαράμετρο “BoW”, που όταν αυτή τεθεί “BoW”(True), τότε από το X_train εξάγονται τα χαρακτηριστικά tfidf των λέξεων και στην συνέχεια αφού πολλαπλασιαστούν καταλληλά με τα σωστά στοιχεία του πίνακα των embeddings, δίνονται στα είσοδο στο νευρωνικό μας μοντέλο.

Έτσι για την ενσωματωση τον BoW χαρακτηριστικών δεν δημιουργείται κάποια νέα κλάση στα ήδη υπάρχοντα μοντέλα μας, αλλά διαφοροποιούνται καταληλα τα αρχικά embeddings που δίνονται σαν είσοδο στο μοντέλο μας ούτως ώστε να αξιοποιηθέι ήδη υπάρχουσα αρχιτεκτονική των μοντέλων. Ο τρόπος αυτός είναι ισοδύναμος με το να δημιουργουσαμε μια νέα κλάση και να κάναμε τον υπολογισμό των συντελεστων tfi-df εσωτερικά, όπου στην συνέχεια θα πολλαπλασιάζαμε τα embeddings με τους κατάλληλους συντελεστές στην διαδικασία “forward”. Απλά επιλέχθηκε ο πρώτος τρόπος, λόγω της εύκολης γενίκευσης και της εφαρμογής σε κάθε ένα από τα μοντέλα μας.

Για την ενσωμαωση των χαρακτηριστικών BoW επιλέγεται το μοντέλο με την καλύτερη επίδοση μέχρι στιγμής. Αυτό το μοντέλο είναι το “BiH_AttentionLSTM”. Έτσι κάνουμε την υπερπαραμετρο BoW = “BoW”, επιλέγουμε σαν μοντέλο το “BiH_AttentionLSTM” και εξάνονται τα ακόλουθα αποτελέσματα για το dataset “MR”:



Από τα παραπάνω αποτελέσματα, διαπιστώνουμε ότι το νεο μοντέλο “BoWBiH_AttentionLSTM” δεν επιτυγχάνει βελτίωση σε σχέση με το προηγούμενο. Το γεγονός αυτό ειναι αναμενόμενο διότι το το dataset που χρησιμοποιείται είναι μεγέθους που δεν επιτρέπει την βελτίωση της αποδοσης μέσω της ενσωμάτωσης των χαρακτηριστικων BoW, όπως θα αναλυθεί και στο επόμενο ερώτημα.

ΕΡΩΤΗΜΑ 6.2 (Bonus)

Η βελτίωση ή μη της απόδοσης που θα προκύψει με την ενσωμάτωση των BoW χαρακτηριστικών στις προκύπτουσες αναπαραστάσεις του νευρωνικού εξαρτάται κατά κύριο λόγω από το μέγεθος του dataset στο οποίο έχει εκπαιδευτεί το μοντέλο. Στην περίπτωση που το dataset είναι σχετικά μικρό, τότε τα BoW χαρακτηριστικά είναι εξειδικευμένα για το συγκεκριμένο dataset με αποτέλεσμα να δίνουν μία πιο ρεαλιστική αναπαράσταση των λέξεων με αποτέλεσμα να οδηγούν σε ορθότερα αποτελέσματα. Σε αντίθετη περίπτωση, όταν το dataset που εργαζόμαστε είναι αρκετά μεγάλο, αυτό έχει σαν αποτέλεσμα οι πίνακες των χαρακτηριστικών που προκύπτουν να είναι αραιοί (μεγάλο πλήθος μηδενικών στοιχείων) με αποτέλεσμα όταν πολλαπλασιάζονται με τις αρχικές αναπαραστάσεις των νευρωνικών (embeddings) να δημιουργούνται εκ νέου αραιοί πίνακες. Έτσι η εκπαίδευση συμβαίνει με βάση αυτούς τους πίνακες και έτσι τα αποτελέσματα που προκύπτουν πάσχουν από σφάλματα, μειώνοντας έτσι την απόδοση του μοντέλου μας.

ΕΡΩΤΗΜΑ 7.1-7.2 (Bonus)

Για τις αναγκες του συγκεκριμένου ερωτήματος, εχρησιμοποιήσαμε την μέθοδο Transfer Learning. Αρχικά, εκαπιδεύσαμε ένα Classifier για το dataset “Semeval 2017 Task4-A” και στη συνέχεια με βάση τα δεδομένα, που δίνονται εκπαιδεύσαμε το RegressionLSTM.

Όπως, είχε αναφερθεί εκπαιδεύσαμε 4 διαφορετικά μοντέλα, ένα για κάθε συναίσθημα. Δημιοργήσαμε κατάλληλες συναρτήσεις για τη φόρτωση και το tokenize των documents και στη συνέχεια ορίσαμε την κλάση RegressionDataset(). Η συγκεκριμένη κλάση δέχεται ως είσοδο τις tokenized προτάσεις, τις λέξεις των συναισθημάτων(anger, fear, joy, sadness), των οποίων η χρήση θα εξηγηθεί παρακάτω, και τα scores, τα οποία αποτελούν δείκτη έντασης του εκάστοτε συναισθήματος σε μια πρόταση.

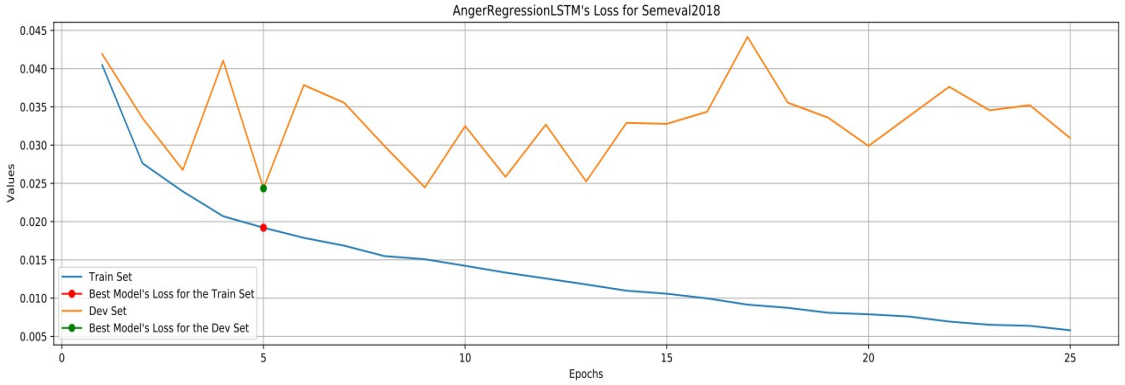
Η __getitem__() επιστρέφει τα διανύσματα των αλέξεων της πρότασης, το score , το length και μια ακόμη παράμετρο emotion, η οποία θα εξηγηθεί αργότερα.

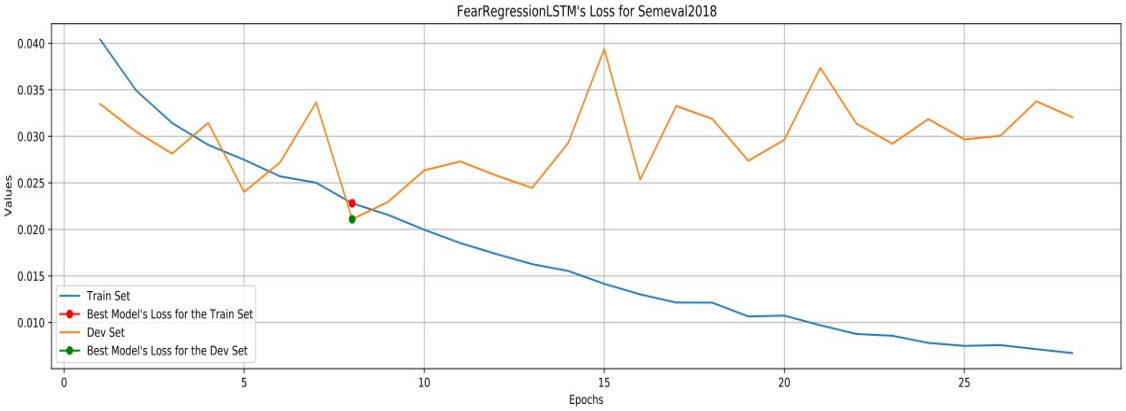
Η αρχιτεκτονική του μοντέλου φαίνεται στην παρακάτω εικόνα.

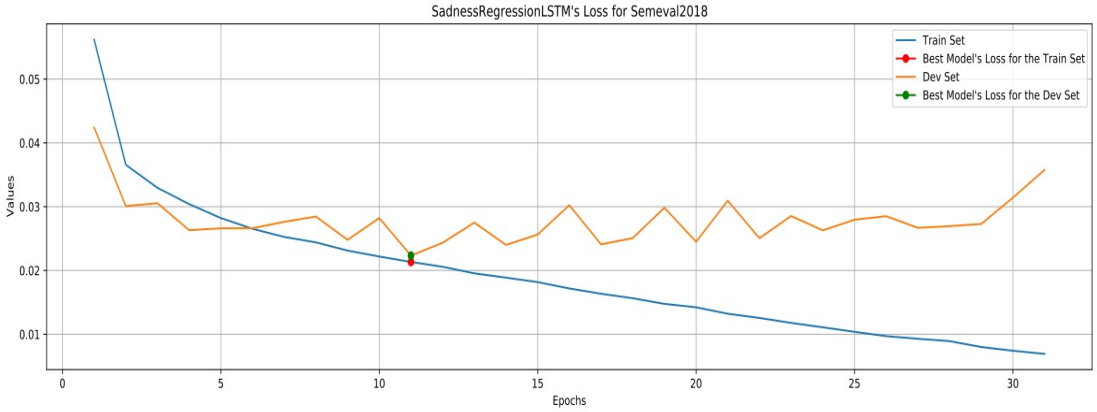
```
LSTMRegression(
  (softmax): Softmax(dim=-1)
  (non_linearity): Tanh()
  (embedding): Embedding(1193516, 50)
  (lstm): LSTM(50, 50, bidirectional=True)
  (outlayer): Linear(in_features=100, out_features=1, bias=True)
)
```

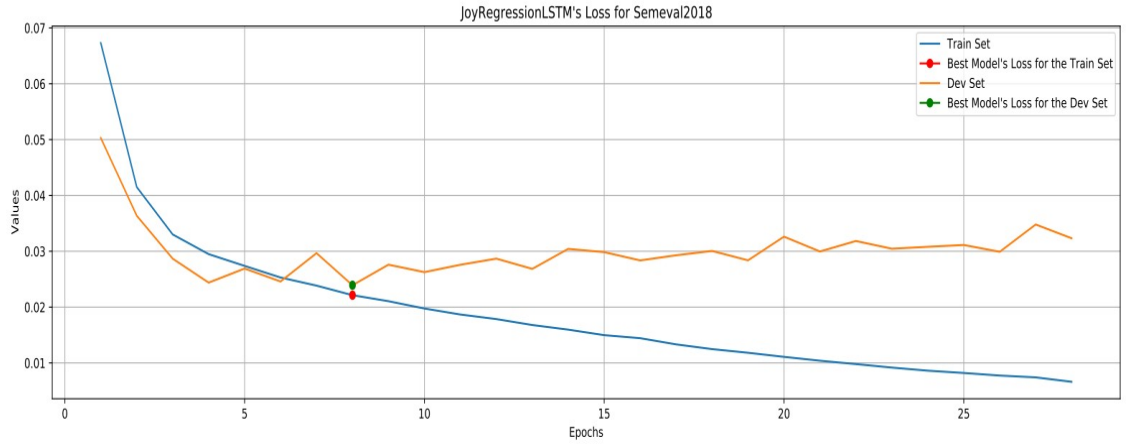
Όπου τα βάρη του Embedding layer είναι freeze. Δοκιμάστηκαν και να τεθούν ως trainable, αλλά δεν υπήρξαν αξιόλογα αποτελέσματα.

Χρησιμοποιώντας το MSELoss(), Adam και φορτώνοντας το προεκπαιδευμένο classifier στο Regression μοντέλο προέκυψαν τα παρακάτω αποτελέσματα.









Σχόλια:

1. Μπορεί να παρατηρήσει κανείς την πτώση του loss για το train set για όλα τα μοντέλα και ταυτόχρονα το overfitting του test set. Ήταν αναμενόμενο, καθώς το dataset στο οποίο εκπαιδεύεται το μοντέλο είναι πολύ μικρό και δυσκολεύεται να γενικεύσει.

Βελτιστοποιήσεις:

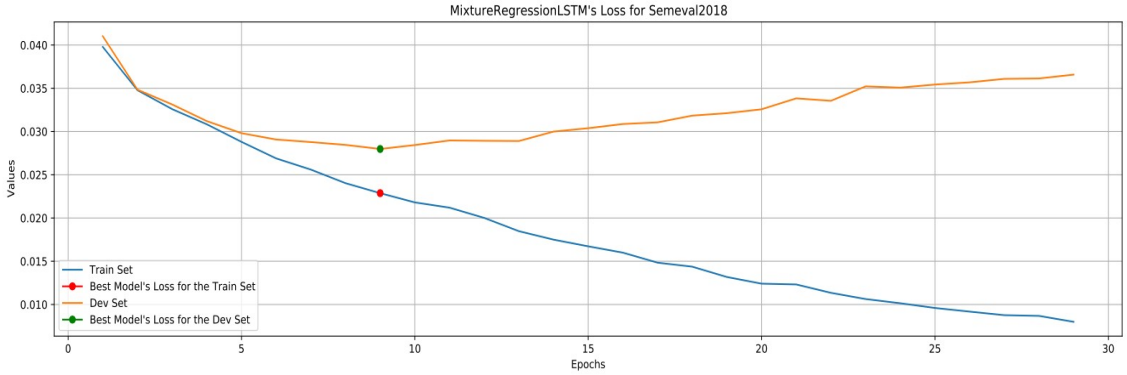
1. Ένα opitmization θα ήταν η χρήση του αλγορίθμου cross-validation, ο οποίος συνίσταται για μικρά dataset, όπως αυτό.
2. Επίσης, η εκπαίδευση του classifier με πολυδιάστατα διανύασματα ενδέχεται να βελτίωνε το μοντέλο.

Η δική μας Ιδέα:

Έχοντας ασχοληθεί πάνω σε τόσες τεχικές, είχαμε την εξής ιδέα. Αντί να φτιάξουμε ένα μοντέλο για κάθε συναίσθημα, κατασκευάσαμε ένα μοντέλο για όλα τα συναισθήματα. Η μόνη διαφορά είναι οι παράμετροι, στους οποίου έγινε λόγος πριν. Έτσι, λοιπόν, η λέξη-παράμετρος (fear, anger, joy, sadness), που δίνεται στο RegressionDataset, αξιοποιείται ως εξής.

1. Βρίσκουμε το αντίστοιχο index από το word2idx
2. Το επιστρέφουμε μέσω της παραμέτρου emotion
3. Στην forward του μοντέλου καλούμε το Embedding layer, για να προσεπλάσουμε την αντίστοιχη λέξη.
4. Τελικά, κάνουμε concat αυτό το διάνυσμα και την έξοδο του LSTM layer, με σκοπό να προσδώσουμε στο μοντέλο μας όσο περισσότερη πληροφορία δύνανται.

Με αυτή την αρχιτεκτονική καταλήξαμε στα εξής αποτελέσματα:



Σχόλιο:

1. Παρατηρούμε ότι, αντιμετωπίζει το ίδιο πρόβλημα overifitting με τα υπόλοιπα, λόγω του μικρού dataset.
2. Επίσης, πετυχαίνει ένα πολύ καλό score, καθώς μειώνει το loss στην ίδια τάξη μεγέθους με τα 4 διαφορετικά μοντέλα.
3. Έχει υλοποιηθεί reguralization με EarlyStopping.

Υποσημείωση:

- Επισυνάπτονται αρχεία .txt, τα οποία αναφέρουν τα στατιστικά των καλύτερων μοντέλων, για τα ερωτήματα 2-4 και ξεχωριστό αρχείο για το ερώτημα 7 στο φάκελο “data”.

ΤΕΛΟΣ 3ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΝΑΦΟΡΑΣ