(Homework 3 / Vaccine Sentiment Classifier using bidirectional stacked RNN with LSTM/GRU)

Ονοματεπώνυμο: Δημήτριος Σιταράς
Αριθμός μητρώου: 1115201800178
Εξάμηνο: 7ο

Περιεχόμενα

Vaccine Sentiment Classifier using bidirectional stacked RNN with LSTM/GRU

Γενικά

Παρακάτω παρουσιάζω το καλύτερο μοντέλο που κατάφερα να υλοποιήσω στο Colab μετά από πολλούς πειραματισμούς που αναλύονται στην συνέχεια.
Αρχικά, εφαρμόζω στα train και validation sets το απαραίτητο preprocessing (cleaning και lemmatize, χρησιμοποιώντας τις ίδιες συναρτήσεις με τις προηγούμενες εργασίες), προτού τα tweets «σπάσουν» σε λέξεις και μετατραπούν σε διανύσματα με την βοήθεια της βιβλιοθήκης torchtext. Συγκεκριμένα, το vectorization των αντίστοιχων sets γίνεται με χρήση των embeddings του Glove, επέλεξα λοιπόν να χρησιμοποιήσω το αρχείο glove.twitter.27B.100d. Δοκιμάζοντας να χρησιμοποιήσω αρχεία με περισσότερες διαστάσεις (200d) καθιστούσαν την ολη διαδικασία πιο χρονοβόρα χωρις να παρατηρώ καλύτερα αποτελέσματα, από την άλλη τα αρχεία με λιγότερες διαστάσεις (25d,50d) είχαν χειρότερες μετρήσεις, λόγω των μειωμένων features που έχουν. Μάλιστα, η χρήση του glove.twitter.27B πέρα από προφανείς λόγους (τα αντίστοιχα sets μας περιέχουν tweets) αποδείχτηκε επίσης η πιο αποδοτική επιλογή, καθώς σε σύγκριση με άλλα αρχεία, όπως glove.6B, τα αποτελέσματα ήταν καλύτερα.
Τελικά, χρησιμοποιώντας τα embedding vectors του Glove, κατασκευάζω ένα λεξικό διανυσμάτων το οποίο σχηματίζεται με βάση τις 10000 πιο δημοφιλής λέξεις των tweets, που εμφανίζονται τουλάχιστον 6 φορές σε αυτά. Πειραματίστηκα να σχηματίσω το λεξικό χωρίς περιορισμούς όσων αναφορά τις λέξεις, το λεξικό που πρόκυπτε ήταν αρκετά μεγάλο, δίνοντας έτσι παραπάνω πληροφορία στο μοντέλο (όσον αναφορά τα embeddings), στην τελική αυτή η προσέγγιση δεν μου έδινε τόσο καλά αποτελέσματα, ίσως επειδή το μοντέλο «μπερδευόταν».
Στη συνέχεια, δημιουργώ 2 iterators από τα train και validation sets αντιστοίχως μέσω του BucketIterator.splits, τα οποία χρησιμοποιούνται για την εκπαίδευση και για την αξιολόγηση του μοντέλου μου. Επέλεξα να ορίσω το batch size ίσο με 100, καθώς δοκιμάζοντας (κυρίως) μεγαλύτερα αλλά και μικρότερα μεγέθη τα αποτελέσματα που παρατηρούσα δεν ήταν τα ιδανικά.
Με βάση την θεωρία, όσο μεγαλύτερο είναι το batch size τόσο πιο λίγο θόρυβο προκαλεί στα gradients και τόσο καλύτερη γίνεται η εκτίμηση, βέβαια αυτό δεν είναι κανόνας, καθώς στην πράξη παρατήρησα ότι το μοντέλου μου συμπεριφέρεται διαφορετικά από το αναμενόμενο που περιγράφει η θεωρία, όπως προανέφερα. Οπότε, τελικά η επιλογή του καλύτερου batch size για ένα μοντέλο προκύπτει μέσα από πειραματισμό και εξαρτάται κυρίως από: τα datasets, το νευρωνικό και την αντικειμενική συνάρτηση (loss function).

Νευρωνικό Δίκτυο

Όσον αναφορά το νευρωνικό δίκτυο που υλοποίησα, σε πρώτο στάδιο, υπάρχει η δυνατότητα να χρησιμοποιηθεί είτε LSTM είτε GRU. Και στις 2 αυτές περιπτώσεις εφόσον ζητείται από την εκφώνηση ("bidirectional stacked RNN") έχω τουλαχιστον 2 layers και την αντίστοιχη παράμετρο bidirectional ορισμένη ως True. Ακολούθως, συμπεριέλαβα ένα Embedding layer προκειμένου να «φορτώσω» σε αυτό τα embeddings που κατασκεύασα προηγουμένως με το Glove, αυτό ουσιαστικά το καταφέρνω μέσω της συνάρτησης pretrained, η οποία μου εξασφαλίζει ότι τα embeddings δεν θα γίνονται retrain (εφόσον η παράμετρος freeze είναι ορισμένη ως True). Ακόμα, πρόσθεσα ένα Linear Layer για την τελική έξοδο, ο αριθμός του hidden dimension που δέχεται ως παράμετρο είναι 2 φορές το αρχικό hidden dimension διότι το μοντέλο είναι bidirectional. Τέλος, πρόσθεσα και ένα Dropout Layer, ώστε να αποτρέψω το overfitting (αυτό το πετυχαίνει αγνοώντας τους τυχαία επιλεγμένους νευρώνες κατά τη διάρκεια της εκπαίδευσης). Σημειώνω πως δεν χρησιμοποιώ output activation function (δοκίμασα την Softmax καθώς συστήνεται για multi-class classification, όμως οι μετρήσεις ήταν χειρότερες), διότι έχω ως loss function την CrossEntropy, η οποία περιλαμβάνει την Softmax.

Hyperparameters

Για το μοντέλο έχω επιλέξει να ορίσω τα εξής hyperparameters:

Hidden Dimension / Size = 40, πειραματίστηκα με μεγαλύτερα μεγέθη (από 50 μέχρι 512) όπου παρατήρησα πως η σύγκλιση καθυστερούσε αρκετά (όσο αυξανόταν το μέγεθος τόσο περισσότερο) έχοντας τελικά τα ίδια αποτελέσματα, με μικρότερα μεγέθη αν και η σύγκλιση ήταν πιο γρήγορη παρατηρούσα μεγαλύτερο underfitting, έτσι που το μοντέλο χρειαζόταν παραπάνω epochs για να εκπαιδευτεί.

Number of Layers / n_layers = 2, μπορούν να είναι το λιγότερο 2 διότι το μοντέλο είναι "bidirectional stacked RNN". Δοκίμασα, λοιπόν, μέχρι και 8 layers, καταλήγοντας σε αυτήν μου την επιλογή καθώς οι μετρήσεις ήταν καλύτερες.

Dropouts = 0.1, πειραματίστηκα με τιμές έως και 0.7, οι οποίες μου εμφάνιζαν διαρκώς underfitting, έτσι που το μοντέλο χρειαζόταν όλο και παραπάνω epochs για να εκπαιδευτεί.

Learning Rate / lr = $2 \cdot 10^{-3}$, καθώς για μεγαλύτερες τιμές επιταχυνόταν η εκμάθηση κατά την εκπαίδευση με αποτέλεσμα το μοντέλο να παρουσίαζε overfit και αστάθεια στα αποτελέσματα, ενώ για μικρότερες τιμές το μοντέλο χρειαζόταν περισσότερα epochs για να εκπαιδευτεί.

Epochs = 11, καθώς αρκούν για να εκπαιδευτεί το μοντέλο, αν και από την αντίστοιχη loss curve παρατηρώ ότι η εκπαίδευση πρέπει να σταματήσει στην 9η εποχή (early stopping) καθώς μετά από αυτή το μοντέλο παρουσιάζει overfitting (αρχίζει να

αυξάνεται το validation loss συγκριτικά με το train loss το οποίο συνεχίζει να μειώνεται).

Batch Size = 100, αναφέρθηκα σε αυτό προηγουμένως.

Cell Type (LSTM / GRU) = LSTM, δοκίμασα και τα 2 (με τις παραπάνω υπερπαραμέτρους), παρατηρώντας πως παίρνω σχεδόν τα ίδια αποτελέσματα. Βέβαια, το GRU, επειδή είναι λιγότερο περίπλοκο από το LSTM, στις δοκιμές που έκανα ήταν αντιληπτά λίγο πιο γρήγορο. Τελικά, επέλεξα να χρησιμοποιήσω το LSTM στο μοντέλο μου, καθώς οι αντίστοιχες loss και ROC curves ήταν ελάχιστα βελτιωμένες συγκριτικά με το GRU (η loss curve έχει λιγότερες «αναταραχές» και η ROC curve «φτάνει υψηλότερα»).

Loss Function

Χρησιμοποιώ την CrossEntropy (όπως ζητείται και στην εκφώνηση).

Optimizer

Χρησιμοποιώ τον Adam (όπως ζητείται και στην εκφώνηση).

Gradient Clipping

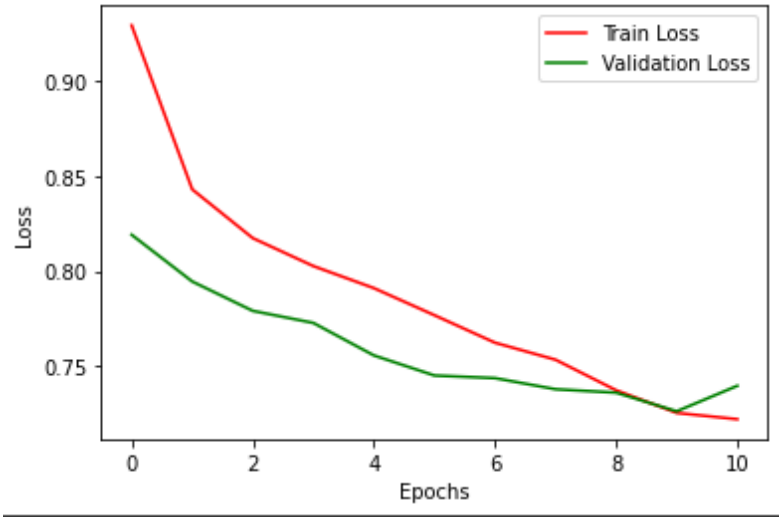
Κατά την διάρκεια της εκπαίδευσης χρησιμοποιώ Gradient Clipping προκειμένου να αποτρέψω «την εξαφάνιση των gradients», έχω δοκιμάσει διάφορες τιμές για το clipping στο εύρος [5,100] χωρίς να παρατηρηθεί κάποια ιδιαίτερη διαφορά, τελικά επέλεξα την τιμή 5. Σε δοκιμές όπου δεν χρησιμοποίησα Gradient Clipping μου αυξήθηκαν υπερβολικά τα train και τα validation losses, σε βαθμό που το μοντέλο να αδυνατούσε να εκπαιδευτεί.

Metrics

Accuracy	65.55 %
F1-score	67 %
Precision	70 %
Recall	66 %

Loss curve

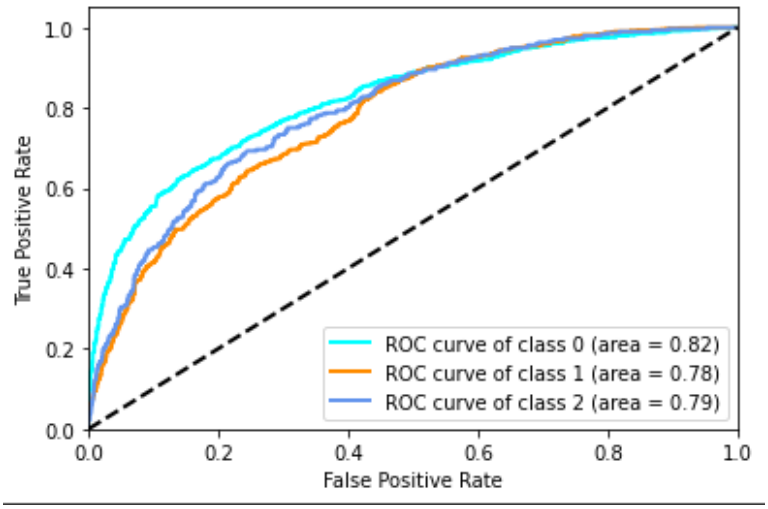
Όσον αναφορά τώρα την loss curve που προκύπτει από τα loss του train και του validation set για το συγκεκριμένο μοντέλο, παρατηρώ πως αρχίζει να κάνει overfitting μετά την 9η εποχή (αρχίζει να αυξάνεται το validation loss συγκριτικά με το train loss, το



οποίο συνεχίζει να μειώνεται). Αυτό σημαίνει πως πρέπει να σταματήσει εκεί η εκπαίδευση (early stopping).

ROC curve

Η ROC δείχνει την απόδοση ενός μοντέλου ταξινόμησης προβάλλοντας την διαγνωστική ικανότητά του. Συνεπώς, όσο πιο κοντά είναι οι καμπύλες της γραφικής παράστασης στην πάνω αριστερή γωνία τόσο μεγαλύτερο True Positive Rate έχει το μοντέλο. Έχω συγχωνεύσει, λοιπόν, 3 ROC curves σε ένα γράφημα για το συγκεκριμένο «πρόβλημα» (multiclass classification), κάθε καμπύλη αντιστοιχεί και επομένως αντιπροσωπεύει κάθε κλάση (0,1,2). Παρατηρώντας, λοιπόν, το γράφημα του μοντέλου μου συμπεραίνω πως αποδίδει καλύτερα για τα tweets που ανήκουν στην κλάση 0 (neutral) και χειρότερα αυτά που ανήκουν στις υπόλοιπες 2 κλάσεις (anti-vax και vax) με μικρή απόκλιση.



Παρατηρήσεις

Με βάση το report της προηγούμενης εργασίας, το καλύτερο μοντέλο από τα Homeworks 1 και 2 είναι αυτό από το Homework 1 (SoftMax regression) που δεν έχει δεχθεί preprocessing και έγινε vectorization με TF-IDF.
Συγκρίνοντας, λοιπόν, αυτά τα 2 μοντέλα παρατηρώ πως κανένα δεν παρουσιάζει underfit ή overfit. Διαφέρουν όμως ως προς τα score των metrics:

	Hw1 (SoftMax Regression)	**Hw3 (bidirectional stacked RNN with LSTM/GRU)**
Accuracy	71.12 %	65.55 %
F1-score	71 %	67 %
Precision	71 %	70 %
Recall	71 %	66 %

είναι επομένως προφανές πως το μοντέλο της πρώτης εργασίας για τα ίδια sets αποδίδει καλύτερα, κάνοντας καλύτερη κατηγοριοποίηση των tweets.

Αξίζει να σημειωθεί επιπλέον, πως το LSTM model έχει αρκετά καλύτερη απόδοση συγκριτικά με το καλύτερο FNN μοντέλο της Homework 2, καθώς πέρα από τα score των metrics που το FNN μοντέλο υστερεί ελάχιστα, το LSTM μοντέλο καταφέρνει να πετύχει σύγκλιση σε μόλις 9 epochs, σε αντίθεση με το FNN το οποίο χρειάζεται 100 epochs. Βέβαια, αυτό είναι λογικό διότι το LSTM μοντέλο έχει πολύ πιο πολύπλοκη δομή από το FNN.

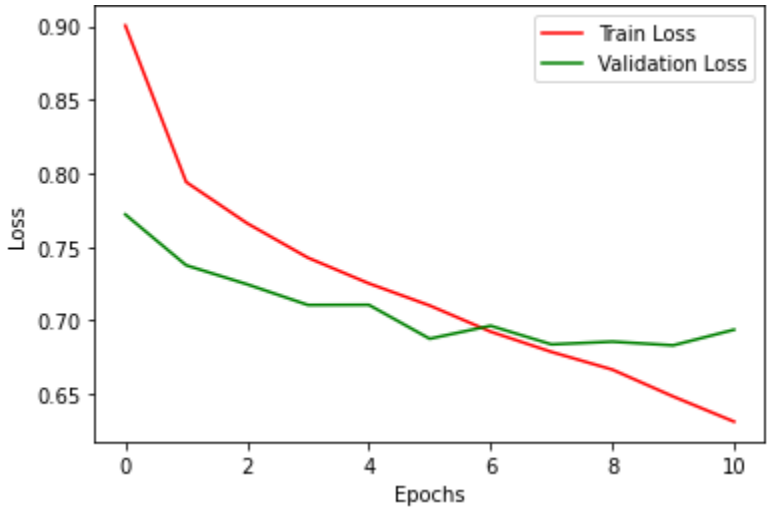
With Attention

Κατάφερα να προσθέσω Attention στο παραπάνω μοντέλο, με την βοήθεια του pdf από το Stanford (το οποίο δόθηκε σε post στο piazza), υλοποιώντας ουσιαστικά «τις εξισώσεις» που αναφέρονται στην σελίδα 75 του pdf. Προκειμένου να χρησιμοποιηθεί το Attention, πρέπει να τεθεί σε True το αντίστοιχο flag (attention_flag) κατά την αρχικοποίηση του μοντέλου.

Με την προσθήκη του παρατήρησα λοιπόν πως, αν και καθυστερεί αρκετά την διαδικασία εκπαίδευσης, βελτιώνει τα αποτελέσματα του μοντέλου, και αυτό γίνεται φανερό από τις μετρήσεις που παρουσιάζονται παρακάτω συγκριτικά με πριν.

	Without Attention	**With Attention**
Accuracy	65.55 %	69.36 %
F1-score	67 %	70 %
Precision	70 %	69 %
Recall	66 %	72 %

Στην συνέχεια, από την loss curve, βλέπω πως αρχίζει να κάνει overfitting μετά την 6η εποχή. Αυτό σημαίνει πως πρέπει να σταματήσει εκεί η εκπαίδευση, καθώς έχει επιτευχθεί σύγκλιση (early stopping στην 6η). Συγκριτικά με πριν την προσθήκη του Attention, το μοντέλο πλέον χρειάζεται πιο λίγες epochs ώστε να εκπαιδευτεί, καθώς προηγουμένως η σύγκλιση επιτυγχανόταν στην 9η epoch.



Τέλος, παρατηρώντας την ROC curve μετά την προσθήκη του Attention συμπεραίνω πως όντως βελτιώθηκε η διαγνωστική ικανότητα του μοντέλου, καθώς σε σχέση με πριν, οι καμπύλες μετατοπίστηκαν, έστω και ελάχιστα, πιο κοντά στην πάνω αριστερή γωνία.