|  |  |  |  |
| --- | --- | --- | --- |
|  | Τεχνητή Νοημοσύνη 2η Εργασία | |  |
|  | |
|  | |
|  | Ταξινόμηση κριτικών IMDB | |  |
| * Σοφία-Ζωή Σωτηρίου : 3210192 * Ευάγγελος Λευτάκης : 3200093 * Ρέα Σκλήκα : 3210181 |  |  |
|  | |  |

Αρχικοποίηση Δεδομένων

Αφού λάβουμε το imdb dataset μαζί το λεξικό word index του μέσω του Keras, προσθέτουμε στο λεξικό τις λέξεις [pad], [bos], [oov] που είναι tokens με ειδική σημασία για την αναπαράσταση κειμένου. Έπειτα, με την χρήση του countVectorizer φιλτράρουμε τις λέξεις και κρατάμε μόνο αυτές που εμφανίζονται τουλάχιστον 100 φορές (min\_df) και παράλληλα μετατρέπουμε τις προτάσεις σε δυαδικές αναπαραστάσεις από 0 για απουσία και 1 για εμφάνιση μιας λέξης σε κάθε πρόταση. Αλλάζουμε τον τύπο τον δεδομένων από float64 που είναι το default σε integer για βελτιωμένη απόδοση. Τέλος, μετατρέπουμε τα δεδομένα μάθησης και ελέγχου σε dense arrays με την χρήση της toarray() καθώς by default λαμβάνουμε ένα sparse array από τον count vectorizer η οποία μορφή δεν είναι συμβατή με πολλές από τις διαδικασίες που απαιτούνται για τις διάφορες μεθόδους τις εργασίας.

A computer screen shot of text

Description automatically generated

Υλοποίηση Καμπυλών και Πινάκων

Για τη παρουσίαση των αποτελεσμάτων των ταξινομητών δημιουργήσαμε συναρτήσεις που συλλέγουν δεδομένα για τα αποτελέσματα και δημιουργούν καμπύλες μάθησης για το train και το validation set, πίνακα με το accuracy των train set και validation set ανάλογα με το μέγεθος του training set και καμπύλες και πίνακα για recall, precision και f1 scores ανάλογα με το μέγεθος του training set.

Περιλαμβάνονται οι συναρτήσεις:

* My\_learning\_curve(estimator, x, y, train\_sizes, scoring, title, cv):

Παίρνει ως παραμέτρους τον ταξινομητή τον οποίο αξιολογεί, τα παραδείγματα εκπαίδευσης και τα αποτελέσματά τους, τα μεγέθη των παραδειγμάτων εκπαίδευσης από τα οποία θα εξάγει συμπεράσματα, τον τύπο της αξιολόγησης (εδώ accuracy), τον τίτλο του διαγράμματος και τον τύπο cross-validation splitting που θα υλοποιηθεί (εδώ πλήθος φορών που θα «διπλωθούν» τα δεδομένα). Με τη χρήση της μεθόδου learning curve της βιβλιοθήκης sklearn.model\_selection παίρνουμε τα accuracy scores του ταξινομητή για τα train και validation sets και με τη χρήση της βιβλιοθήκης matplotlib.pyplot δημιουργείται το διάγραμμα των learning curves και με τη pandas δημιουργείται ο πίνακας που παρουσιάζει τα αποτελέσματα.

* custom\_classification\_report(estimator,x\_train, y\_train,x\_test, y\_test, train\_sizes,name):

Παίρνει ως παραμέτρους τον ταξινομητή τον οποίο αξιολογεί, τα δεδομένα εκπαίδευσης και ελέγχου, τα μεγέθη των παραδειγμάτων από τα οποία θα εξάγει συμπεράσματα, και το όνομα του ταξινομητή που αξιολογείται. Καλεί τις συναρτήσεις classification\_diagrams και my\_learning\_curve.

* classification\_diagrams(estimator,x\_train, y\_train,x\_test, y\_test, train\_sizes, name):

Παίρνει τις ίδιες παραμέτρους με τη custom\_classificcation\_report. Για κάθε μέγεθος δεδομένων του train\_sizes καλεί τις μεθόδους fit και predict του ταξινομητή και με βάση τις προβλέψεις που γίνονται κάθε φορά καλεί τις συναρτήσεις precision\_score, recall\_score και f1\_score τα αποτελέσματα των οποίων τοποθετεί σε arrays ανάλογα με τον τύπο του score (precision, recall ή f1). Όπως και η my\_learning\_curve, με συναρτήσεις των matplotlib.pyplot και pandas δημιουργεί τα διαγράμματα και τον πίνακα αντίστοιχα των precision, recall και f1 scores ανά μέγεθος παραδειγμάτων του ταξινομητή.

1ο Μέρος

Για να κάνουμε την κατηγοριοποίηση των κριτικών από το dataset μας στο πρώτο κομμάτι της εργασίας αναπτύξαμε APIs για τους αλγορίθμους μηχανικής μάθησης Naive Bayes, Logisitic Regression και AdaBoost και να αξιολογήσουμε τις επιδόσεις τους.

Logistic regression:

Ο αλγόριθμος που υλοποιήσαμε είναι λογιστικής παλινδρόμησης με L2 κανονικοποίηση και κανόνα ανανέωσης βαρών: 

Η διεπαφή αποτελείται από τις παρακάτω μεθόδους:

* \_\_init\_\_(*epochs* ,*learning\_rate* ,*threshold* ,*regularization\_factor*): Παίρνει ως ορίσματα τις διάφορες υπερπαραμέτρους όπως αριθμός επαναλήψεων, ρυθμός μάθησης παράγοντας κανονικοποίησης και threshold. Κατασκευάζει και επιστρέφει τον ταξινομητή με τα ορίσματα που προσδιορίσαμε.
* fit(x\_train\_input,y\_train\_input,learning\_curve): Παίρνει ως ορίσματα τα δεδομένα εκπαίδευσης και μια boolean τιμή σε περίπτωση που θέλουμε να παραχθεί καμπύλη μάθησης. Αρχικά διαιρούμε τα δεδομένα σε εκπαίδευσης και επικύρωσης 80-20 αντίστοιχα. Για κάθε εποχή τα παραδείγματα αναδιατάσσονται τυχαία προς αποφυγή overfitting και να μην δημιουργηθούν γνώσεις που έχουν να κάνουν με την διάταξη των παραδειγμάτων εκπαίδευσης. Για την ενημέρωση των βαρών χρησιμοποιούμε Stochastic Gradient Ascent χωρίς mini batches. Επιστρέφει διάφορες μετρικές που παράγονται από την evaluate() (εμφανίζεται παρακάτω
* predict(x\_test): Παίρνει ως όρισμα τα δεδομένα ελέγχου. Με τον τύπο τις σιγμοειδούς καμπύλης υπολογίζουμε την πιθανότητα κάθε παράδειγμα ελέγχου να είναι στην θετική κατηγορία. Έπειτα αν η πιθανότητα αυτή είναι μεγαλύτερη από το threshold θωρούμε ότι η κριτική είναι θετική. Επιστρέφει έναν πίνακα y\_predicted με μήκος ίσο με τα παραδείγματα ελέγχου ο οποίος αποτελείται από 0 και 1 για εκτιμόμενα θετική ή αρνητική κριτική.
* predict\_prob(x\_test): Το ίδιο με την predict απλά επιστρέφει τις εκτιμώμενες πιθανότητες για τα δεδομένα ελέγχου χωρίς να τα κατατάσσει.
* initialize\_weights(): Αρχικοποιούμε τα βάρη σε τυχαίους αριθμούς πολύ κοντά στο μηδέν.
* update\_weights(x\_train, y\_train): Χρησιμοποιώντας sga με τον τύπο ανανεώνουμε τα βάρη με τον τύπο ανανέωσης.
* evaluate(y\_test, y\_predicted): Συγκρίνουμε τα αποτελέσματα που εκτιμήθηκαν με τα πραγματικά και επιστρέφουμε ένα πίνακα με μετρικές όπως [precision,recall,f1]
* get\_params(): Υλοποιούμε για συμβατότητά με μεθόδους του sci-kit learn όπως learning\_curve()
* classification\_diagrams(x\_train, y\_train, x\_test, y\_test, train\_sizes): Παράγει τρία plots για την πορεία των precision, recall, f1 score συναρτήσει του μεγέθους των δεδομένων εκπαίδευσης.
* custom\_classification\_report(*x\_train*, *y\_train*,*x\_test*, *y\_test*, *train\_sizes*): Καλεί την classification\_diagrams() και την my\_learning\_curve() για να δώσει μια πλήρη εικόνα των αποτελεσμάτων μάθησης του ταξινομητή που π.
* Άλλες Βοηθητικές μέθοδοι όπως set\_threshold() για αλλαγή του threshold ενός ταξινομητή, sigmoid() για υπολογισμό του τύπου .

Αποτελέσματα:

Ο ταξινόμητής που χρησιμοποιήσαμε για τα δεδομένα μάθησης τελικά είχε τις ύπερ παραμέτρους:

* Threshold: του οποίου δώσαμε διάφορες τιμές και καταλήξαμε ότι κάτι κοντά στο (0.4,0.6) έχει καλά αποτελέσματα και μια καλή ισορροπία ανάμεσα σε precision και recall.
* Regularization Factor λ: βάζουμε χαμηλό παράγοντα κανονίκοποίησης (0.001) καθώς για το πρόβλημα μας που είναι sentiment analysis επωφελούμαστε από την καλή αφομοίωση των δεδομένων εκπαίδευσης και αποφεύγοντας το overfitting
* Learning Rate: Αφήνουμε την τιμή στο default που έχουμε ορίσει 0.001 που προσφέρει πιο αργό ρυθμό μάθησης αλλά περιορίζει το ενδεχόμενο να χάσουμε κατά πολύ τα βέλτιστα βαρύ και να ταλαντευόμαστε γύρω από την μέγιστη τιμή πιθανοφάνειας
* Epochs: Επιλέξαμε 100 εποχές κρίνοντας από τα αποτελέσματα της καμπύλης μάθησης.

A graph of a logistic regression results

Description automatically generated A screenshot of a computer

Description automatically generated

Στην καμπύλη μάθησης παρατηρούμε την μείωση της ακρίβειας στα δεδομένα εκπαίδευσης που συνάδει με την μείωση του overfitting, ενώ αυξάνεται η ακρίβεια στα Development δεδομένα που δείχνει ότι ο αλγόριθμος μαθαίνει.

A graph of a logistic regression learning curve

Description automatically generated A screenshot of a graph

Description automatically generated

Naïve Bayes:

Ο αλγόριθμος που χρησιμοποιείται είναι αυτός του Αφελή ταξινομητή Bayes (πολυμεταβλητή μορφή Bernoulli).

Η διεπαφή αποτελείται από τις παρακάτω μεθόδους:

* \_\_init\_\_(alpha): Παίρνει ως όρισμα την υπερπαράμετρο alpha, η οποία είναι η εκτιμήτρια Laplace και κατασκευάζει τον ταξινομητή αρχικοποιώντας τη πιθανότητα ένα παράδειγμα να είναι θετικό (και από εκεί προκύπτει και η πιθανότητα να είναι αρνητικό ως 1-θετικήΠιθανότητα) και οι δεσμευμένες πιθανότητες αν μία λέξη δεν υπάρχει | το παράδειγμα είναι θετικό και αν μια λέξη δεν υπάρχει | το παράδειγμα είναι αρνητικό. Από αυτές τις δύο προκύπτουν και οι δεσμευμένες πιθανότητες αν υπάρχει μια λέξη το παράδειγμα να είναι θετικό/αρνητικό.
* fit(x\_train\_input,y\_train\_input): Παίρνει ως ορίσματα τα δεδομένα εκπαίδευσης και αρχικά τα χωρίζει (80% για εκπαίδευση και 20% για επικύρωση). Έπειτα υπολογίζει τη πιθανότητα ένα παράδειγμα να είναι θετικό με βάση τα δεδομένα εκπαίδευσης. Μετά για κάθε λέξη υπολογίζει τη δεσμευμένη πιθανότητα αν αυτή η λέξη δεν υπάρχει | το αποτέλεσμα είναι θετικό/αρνητικό χρησιμοποιώντας και εκτιμήτρια Laplace.
* getParams(): Επιστρέφει τη παράμετρο alpha.
* predict(x\_test): Παίρνει ως όρισμα τα δεδομένα ελέγχου. Για κάθε παράδειγμα υπολογίζονται τα αθροίσματα των λογαρίθμων όλων των δεσμευμένων πιθανοτήτων κάθε λέξης του παραδείγματος και για θετικό και για αρνητικό αποτέλεσμα. Προστίθενται και οι λογάριθμοι των πιθανοτήτων ένα παράδειγμα να είναι θετικό/αρνητικό και συγκρίνονται τα δύο αθροίσματα. Επιστρέφει έναν πίνακα με μήκος ίσο με τα παραδείγματα ελέγχου ο οποίος αποτελείται από 0 και 1 για εκτιμόμενα θετική ή αρνητική κριτική ανάλογα με το ποιο άθροισμα ήταν μεγαλύτερο.
* predict\_proba(x\_test): Κάνει την ίδια δουλειά με το predict απλά επιστρέφει μόνο τις πιθανότητες που υπολόγισε χωρίς να κατατάσσει τα παραδείγματα.
* evaluate(y\_true, y\_predicted): Συγκρίνει τα αποτελέσματα της εκτίμησης με τα σωστά αποτελέσματα και επιστρέφει έναν πίνακα με μετρικά (precision, recall, f1 score).
* classification\_diagrams(x\_train, y\_train,x\_test, y\_test, train\_sizes): Δέχεται όλα τα δεδομένα εκπαίδευσης και ελέγχου και παράγει διαγράμματα precision, recall, f1 πίνακα με τα σκορ αυτών των μετρικών.
* custom\_classification\_report(x\_train, y\_train,x\_test, y\_test, train\_sizes): Δέχεται όλα τα δεδομένα εκπαίδευσης και ελέγχου και παράγει καμπύλη εκπαίδευσης.

Αποτελέσματα:

Ο ταξινόμητής που χρησιμοποιήσαμε για τα δεδομένα μάθησης τελικά είχε την ύπερ παράμετρο:

* Alpha: η οποία είναι η εκτιμήτρια Laplace που χρησιμοποιήθηκε για να αποφευχθεί η επιρροή που θα είχε η τυχών μηδενική εμφάνιση μιας λέξης στους υπολογισμούς. Η τιμή που της δόθηκε είναι 1 ώστε να είναι η χαμηλότερη δυνατή και τα αποτελέσματα να είναι όσο πιο κοντά στα πραγματικά γίνεται.

A graph of a line graph

Description automatically generated with medium confidenceΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαA black and white screen with numbers

Description automatically generated

AdaBoost:

Για την υλοποίηση του αλγορίθμου AdaBoost χρησιμοποιήσαμε το DecisionTreeClassifier του scikit-learn με βάθος ίσο με 1 σαν weak learner.

Η διεπαφή αποτελείται από τις παρακάτω μεθόδους:

* \_\_init\_\_(amount\_of\_says, models): O κατασκευαστής των αντικειμένων AdaBoost. Παίρνει ως ορίσματα πίνακα models που περιέχει decision stumps και έναν πίνακα με τα amount of say των δέντρων αυτών. Αν δε δοθούν ορίσματα, αρχικοποιούνται ως None.
* get\_params(): Υλοποιούμε για συμβατότητά με μεθόδους του sci-kit learn όπως learning\_curve()
* fit(x\_train\_input, y\_train\_input, M): Παίρνει ως ορίσματα τα δεδομένα εκπαίδευσης και τον αριθμό επαναλήψεων. Αρχικά διαιρούμε τα δεδομένα σε εκπαίδευσης και επικύρωσης 80-20 αντίστοιχα. Έπειτα, αρχικοποιούνται ισάξια βάρη για καθένα από τα παραδείγματα εκπαίδευσης και αρχίζουν οι επαναλήψεις. Σε κάθε επανάληψη, δημιουργείται ένα δέντρο απόφασης βάθους 1 το οποίο εκπαιδεύεται στα δεδομένα και ταξινομεί τα παραδείγματα. Ύστερα, υπολογίζεται το σφάλμα του δέντρου, το amount of say και ενημερώνονται τα βάρη και τα δεδομένα. Αν το σφάλμα είναι μεγαλύτερο από 50%, η επανάληψη επαναλαμβάνεται, αλλιώς τα στοιχεία του δέντρου αποθηκεύονται και συνεχίζουμε με την επόμενη επανάληψη.
* update\_data(x, y): Δέχεται ως ορίσματα τα δεδομένα εκπαίδευσης. Στην αρχή αρχικοποιούμε ένα καινούριο άδειο dataset και δημιουργούμε έναν πίνακα (distribution) ο οποίος σε κάθε i-στή θέση περιέχει το άθροισμα των βαρών μέχρι και το i-στό βάρος. Στη συνέχεια, διαλέγουμε τυχαία έναν αριθμό από το 0 έως το 1 και βρίσκουμε το πρώτο στοιχείο του distribution το οποίο είναι μεγαλύτερο ή ίσο του. Βρίσκουμε σε ποιο παράδειγμα αντιστοιχεί το βάρος αυτό και το προσθέτουμε στην καινούρια βάση. Επαναλαμβάνουμε μέχρι το νέο dataset γεμίσει με δεδομένα. Τέλος, αρχικοποιούμε ξανά ισάξια τα βάρη και επιστρέφουμε τα νέα δεδομένα.
* change\_weights(pred, y, amount\_of\_say): Τα ορίσματα της μεθόδου είναι οι προβλέψεις ενός δέντρου απόφασης, οι πραγματικές τιμές και το amount of say του δέντρου αυτού. Μειώνουμε τα βάρη στα παραδείγματα που το δέντρο αξιολόγησε σωστά και αυξάνουμε τα βάρη στα παραδείγματα που το δέντρο κατάταξε λανθασμένα. Με αυτόν τον τρόπο, στις επόμενες επαναλήψεις δίνεται περισσότερη βαρύτητα στα λανθασμένα δεδομένα. Πριν επιστρέψουμε τα βάρη, φροντίζουμε το άθροισμα τους να είναι ίσο με τη μονάδα.
* predict(x): Δέχεται ως είσοδο ένα σύνολο παραδειγμάτων. Για κάθε δέντρο που δημιουργήθηκε, υπολογίζουμε το γινόμενο του amount of say με την απόφαση του δέντρου για κάθε παράδειγμα. Οι τελικές αποφάσεις είναι ό,τι αποφάσισε η πλειοψηφία των δέντρων για κάθε παράδειγμα.
* evaluate(y\_true, y\_predicted): Οι παράμετροι είναι οι πραγματικές κατηγορίες κάποιων παραδειγμάτων και οι κατηγορίες στις οποίες τα κατάταξε ο αλγόριθμος μας. Υπολογίζει τις τιμές των precision, recall, f1 χρησιμοποιώντας τις μεθόδους του scikit learn.
* classification\_diagrams(x\_train, y\_train, x\_test, y\_test, train\_sizes): Παράγει τρία plots για την πορεία των precision, recall, f1 score συναρτήσει του μεγέθους των δεδομένων εκπαίδευσης.
* custom\_classification\_report(*x\_train*, *y\_train*,*x\_test*, *y\_test*, *train\_sizes*): Καλεί την classification\_diagrams() και την my\_learning\_curve() για να δώσει μια πλήρη εικόνα των αποτελεσμάτων μάθησης του ταξινομητή.

Αποτελέσματα:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, αριθμός

Περιγραφή που δημιουργήθηκε αυτόματα

2ο Μέρος

Σε αυτό το σημείο θα συγκρίνουμε τις υλοποιήσεις των αλγορίθμων μας με τις υλοποιήσεις του sci-kit learn. Για αυτή την σύγκριση θα χρησιμοποιηθούν οι ίδιες καμπύλες και μετρικά όπως και στο προηγούμενο μέρος για να είναι ξεκάθαρη η σύγκριση των αποτελεσμάτων.

Logistic regression:

Καμπύλες μάθησης: (Στα αριστερά η δική μας στα δεξία αυτή της sci-kit learn). Παρατηρούμε παρόμοια πορεία στις δυο καμπύλες μάθησης με την δικιά μας υλοποίηση να φτάνει σε μικρότερη ακρίβεια στα δεδομένα εκπαίδευσης που ίσως σημαίνει λιγότερο overfitting ενώ πετυχαίνει σχεδόν ίση ακρίβεια στα δεδομενα επικύρωσης.

A graph of a logistic regression learning curve

Description automatically generatedA graph of a logistic regression learning curve

Description automatically generated

A screenshot of a graph

Description automatically generatedA screenshot of a graph

Description automatically generated

Τα αποτελέσματα της Λογιστικής Παλινδρόμησης της βιβλιοθήκης sci-kit learn:

A graph of a logistic regression results

Description automatically generatedA screenshot of a computer

Description automatically generated

Τα αποτελέσματα της δικής μας υλοποίησης της Λογιστικής Παλινδρόμησης:

A graph of a logistic regression results

Description automatically generatedA screenshot of a computer

Description automatically generated

Naïve Bayes:

Καμπύλες μάθησης: Όπως και από πάνω, στα αριστερά είναι η δική μας και στα δεξία αυτή της sci-kit learn. Η διαφορά στην απόδοση είναι αισθητή με το training accuracy της sci-kit να πέφτει αρκετά όσο αυξάνονται τα παραδείγματα. Αντίθετα, το validation accuracy της sci-kit δεν μεταβάλλεται τόσο, σε αντίθεση με τη δική μας υλοποίηση, όπου το validation accuracy αυξάνεται απότομα.

A graph of a graph with numbers and lines

Description automatically generated with medium confidenceA graph of training and validation

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated A screenshot of a graph

Description automatically generated

Τα αποτελέσματα του Αφελή Bayes της βιβλιοθήκης sci-kit learn:

A graph of a number of different colored lines

Description automatically generated with medium confidence A screenshot of a graph

Description automatically generated

Τα αποτελέσματα του Αφελή Bayes της δικής μας υλοποίησης:

A graph of a line graph

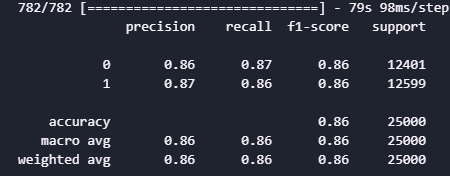
Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated

3ο Μέρος

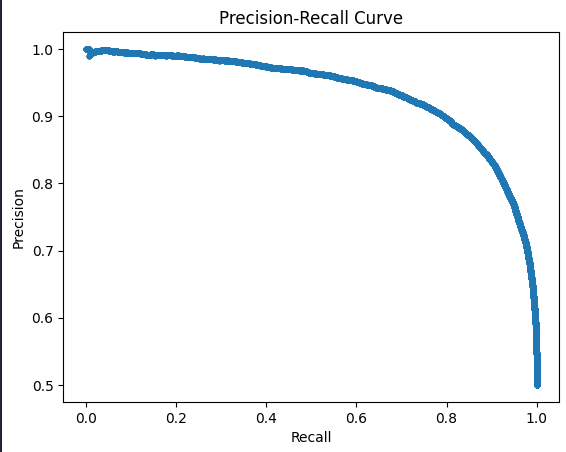
Υλοποιήσαμε ένα stacked Bidirectional GRU RNN με μηχανισμό early stopping και L2 κανονικοποίηση για αποφυγή υπερεφαρμογής. Χρησιμοποιούμε διάφορους αριθμούς εποχών για πειραματισμό όπως 3,5,10 όμως όπως θα δούμε και παρακάτω μετά από μόλις δύο εποχές το μοντέλο αρχίζει να υπερέφαρμόζεται στα δεδομένα. Για συνάρτηση απώλειας χρησιμοποιούμε Cross entropy καθώς έχει καλή ικανότητα να τιμωρεί μεγάλα σφάλματα ενώ αποφεύγει θέματα όπως το vanishing gradient που αντιμετωπίζουν τα rnns. L2 κανονικοποίηση γίνεται με τελεστή 0.0001 και dropout=0.2. Τέλος, χρησιμοποιήσαμε έως 2 biGRU layers καθώς ο χρόνος διογκώθηκε γρήγορα με την αύξηση τον layers κάνοντας την προσαρμογή τον υπερπαραμέτρων δύσκολη.

Αποτελέσματα :



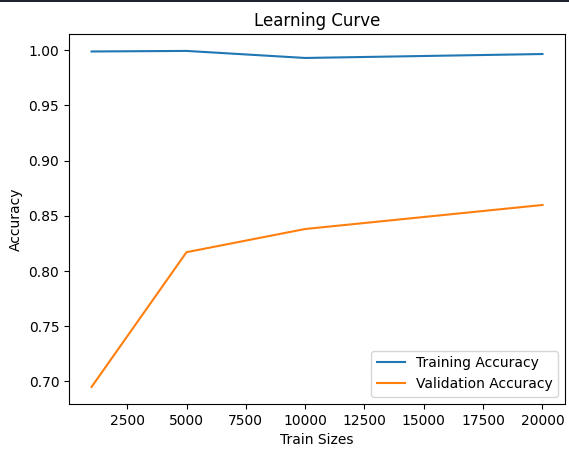
Αυτές είναι οι τιμές για threshold 0.5.

Precision-Recall διάγραμμα:



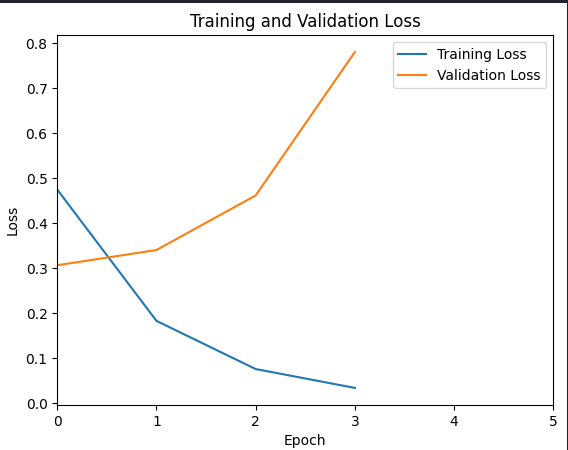
Παρατηρούμε πάρα πολύ υψηλές τιμές Precision και Recall στο test data μεγαλύτερες από όλους τους προηγούμενους αλγορίθμους. Αυτό πιθανό να ευθύνεται στην ικανότητα του biGRU rnn να μάθαίνει το context λέξεων και φράσεων απο επόμενα και προηγούμενα τμήματα κειμένου.

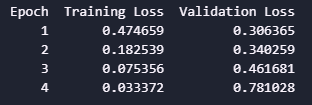
Διάγραμμα learning curve:



Η καμπύλη μάθησης δείχνει σταδιακή αύξηση τις ακρίβειας στα validation δεδομένα με το πλήθος των παραδειγμάτων εκπαίδευσης όπως θα έπρεπε. Ενώ μια μικρή μείωση ακρίβειας στα δεδομένα εκπαίδευσης που είναι πιθανό ενδεικτικό υπερεφρμογής.

Διάγραμμα απώλειας:





Παρατηρούμε ότι και από το τέλος της πρώτης μόλις εποχής η απώλεια στα δεδομένα εκπαίδευσης πέφτει δραματικά ενώ να αυξάνεται στα δεδομένα εκσφαλμάτωσης πράγμα ενδεικτικό υπερεφαρμογής παρά των μηχανισμών αντιμετώπισης που έχουν εφαρμοστεί.

Συνοπτικά το μοντέλο φαίνεται να έχει αρκετά καλή απόδοση και παρόλο που φαίνεται το μοντέλο να υπερεφαρμόζεται στα δεδομένα αρκετά γρήγορα ο μηχανισμός early stopping κρατάει τις καλύτερες τιμές απώλειας για τα δεδομένα εκσφαλμάτωσης .