Προεπεξεργασία Κειμένων

Η προεπεξεργασία κειμένων πραγματοποιείται στο αρχείο preprocess.py.

Βήματα Προεπεξεργασίας:

1. Φόρτωση των δεδομένων
   - Τα δεδομένα προέρχονται από το IMDB Large Movie Review Dataset, που περιέχει κριτικές ταινιών με θετικά (+1) ή αρνητικά (-1) συναισθήματα.
   - Οι κριτικές φορτώνονται από τα αρχεία κειμένου και αποθηκεύονται σε λίστες.
2. Επεξεργασία κειμένου
   - Αφαίρεση ειδικών χαρακτήρων και σημείων στίξης (π.χ., .,!?()).
   - Μετατροπή όλων των χαρακτήρων σε πεζά (lowercase) για συνέπεια στην αναπαράσταση των λέξεων.
   - Αφαίρεση αριθμών και μη αλφαβητικών χαρακτήρων.
3. Διαίρεση σε λέξεις (Tokenization)
   - Χρησιμοποιείται tokenization για τη μετατροπή του κειμένου σε λίστα λέξεων.
4. Αφαίρεση πολύ συχνών και πολύ σπάνιων λέξεων
   - Αρχικά, μετράμε πόσες φορές εμφανίζεται κάθε λέξη στα κείμενα.
   - Έπειτα αφαιρούνται οι n πιο συχνές λέξεις και οι k πιο σπάνιες λέξεις
5. Επιλογή χαρακτηριστικών (Feature Selection) με Information Gain
   - Από τις εναπομείνασες λέξεις, επιλέγονται οι m σημαντικότερες, χρησιμοποιώντας Information Gain.
   - Το Information Gain μετράει πόσο πληροφοριακή είναι μια λέξη για την ταξινόμηση των κριτικών σε θετικές/αρνητικές.
6. Μετατροπή σε διανύσματα χαρακτηριστικών (Feature Vectors)
   - Για τα μοντέλα AdaBoost & Random Forest, κάθε κείμενο αναπαρίσταται ως δυαδικό διάνυσμα όπου αν μια λέξη υπάρχει στο κείμενο αναπαριστάται με 1 και αν δεν υπάρχει με 0.
   - Για το RNN, κάθε κείμενο μετατρέπεται σε σειρά αριθμών που αντιστοιχούν σε λέξεις
7. Padding & Truncation (μόνο για RNN)
   - Truncation: Αν μια κριτική είναι μεγαλύτερη από max_len, κόβουμε τις επιπλέον λέξεις.
   - Padding: Αν είναι μικρότερη, προσθέτουμε μηδενικά (0) στο τέλος.

Random Forest

Υλοποιήσαμε τον αλγόριθμο τυχαίου δάσους (random forest), στον οποίο κατά το στάδιο της εκπαίδευσης κατασκευάζουμε πολλά δέντρα απόφασης, όπου κάθε δέντρο εκπαιδεύεται σε διαφορετική παραλλαγή του συνόλου εκπαίδευσης, χρησιμοποιώντας διαφορετικό υποσύνολο των διαθέσιμων ιδιοτήτων. Ακολουθούμε τη γνώμη της πλειοψηφίας των δέντρων. Έπειτα στο στάδιο του development/validation και του testing για κάθε παράδειγμα ακολουθούμε τη γνώμη της πλειοψηφίας των δέντρων.

Μοντέλο

Ο κώδικας του μοντέλου βρίσκεται το αρχείο randomforest.py.

Το μοντέλο εκπαιδεύεται μέσω της συνάρτησης randomforest_train (εντός του randomforest.py), η οποία δέχεται ως ορίσματα:

- ένα np.ndarray X, όπου είναι ένας πίνακας με τα διανύσματα τα οποία αναπαριστούν τα παραδείγματα εισόδου
- np.ndarray y, όπου είναι ένας πίνακας με τα labels των features
- έναν ακέραιο n_estimators (με προκαθορισμένη τιμή 100), όπου δηλώνει από πόσα δέντρα θα αποτελείται το δάσος
- έναν ακέραιο m_features (με προκαθορισμένη τιμή None), όπου δηλώνει πόσα διαφορετικά features θα χρησιμοποιηθούν για την κατασκευή του κάθε δέντρου
- έναν ακέραιο max_depth (με προκαθορισμένη τιμή None), όπου δηλώνει το μέγιστο βάθος του κάθε δέντρου

Επιστρέφει κατά την ολοκλήρωση της μια λίστα με tuples από αντικείμενα DecisionTreeClassifier και np.ndarray

Αν δεν δοθεί τιμή στην m_features κατά την κλήση της συνάρτησης, τότε δίνεται η ρίζα του πλήθους των features. Έπειτα για κάθε ένα δέντρο δημιουργεί τυχαία μια παραλλαγή του συνόλου εκπαίδευσης (με ίδιο αριθμό παραδειγμάτων) και επιλέγει ένα τυχαίο σύνολο από τα αρχικά features, μεγέθους m_features. Έπειτα δημιουργεί το δέντρο μέσω του DecisionTreeClassifier της Scikit-learn και το προσθέτει στο δάσος.

Η πρόβλεψη της κατηγορίας ενός συνόλου παραδειγμάτων γίνεται μέσω της μεθόδου randomforest_predict (εντός του randomforest.py). Στο τέλος επιστρέφει τις κατηγορίες των παραδειγμάτων. Για κάθε δέντρο δημιουργεί ένα ndarray με την κατηγορίες των παραδειγμάτων, έπειτα ελέγχει (μέσω της συνάρτησης mode) για κάθε παράδειγμα ποια τιμή εμφανίζεται περισσότερες φορές και επιστρέφει αυτή.

Train & Development

Ο κώδικας για την εκπαίδευση του μοντέλου βρίσκεται το αρχείο train_randomforest.py. Οι συναρτήσεις plot_learning_curve_A, plot_learning_curve_B είναι βοηθητικές και δημιουργούν και αποθηκεύουν απλά τα ζητούμενα διαγράμματα. Για κάθε ένα συνδυασμό υπερπαραμέτρων εκπαιδεύει το μοντέλο και το έτοιμο μοντέλο RandomForestClassifier της Scikit-learn για την σύγκριση των αποτελεσμάτων, κάθε φορά που βρίσκει έναν καλύτερο συνδυασμό υπερ παραμέτρων (με κριτήριο το development accuracy) αποθηκεύει τον Custom Random Forest,τον RandomForestClassifier, κατασκευαζει και αποθηκεύει τα διαγράμματα και εμφανίζει τα αποτελέσματα των μετρικών. Στο τέλος εμφανίζει τις καλύτερες υπερπαραμέτρους.

Επιλογή υπερπαραμέτρων:

- n_estimators: Δοκιμή από το σύνολο [10,50,150,200,300,400,500,,700,800,900], οι τιμές > 300 είχαν χαμηλό dev_acc οπότε απορρίφθηκαν. Η καλύτερη ήταν 700.
- m_features: None (άρα τετραγωνική ρίζα). Η παράμετρος αυτή επιλέχθηκε καθώς στο σύγγραμα των Russell & Norvig στην σελίδα 775 αναφέρει ότι για προβλήματα ταξινόμησης επιλέγεται η τετραγωνική ρίζα
- m_values: Δοκιμή από το σύνολο [1000,2000,3000,4000,5000,7000,10000], οι τιμές εκτός των 5000,7000 είχαν αρκετά χαμηλότερο dev_acc οπότε απορρίφθηκαν. Η καλύτερη ήταν 5000.
- n_most_values: Δοκιμή από το σύνολο [20,50,100], η καλύτερη τιμή ήταν 50.

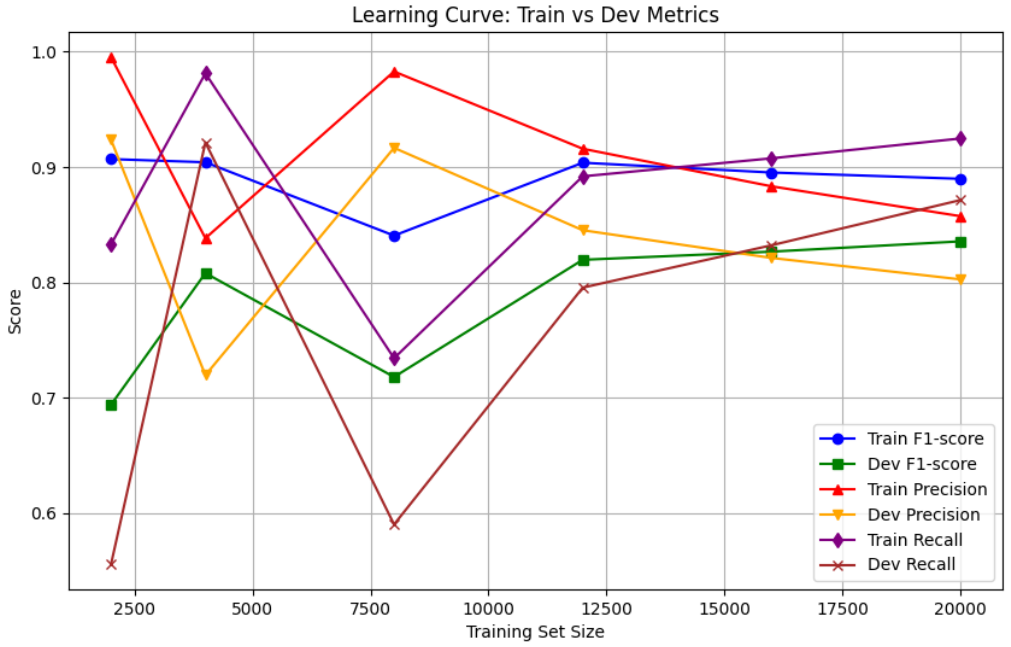
- k_rarest_values: Δοκιμή από το σύνολο [20,50,100], η καλύτερη τιμή ήταν 50.

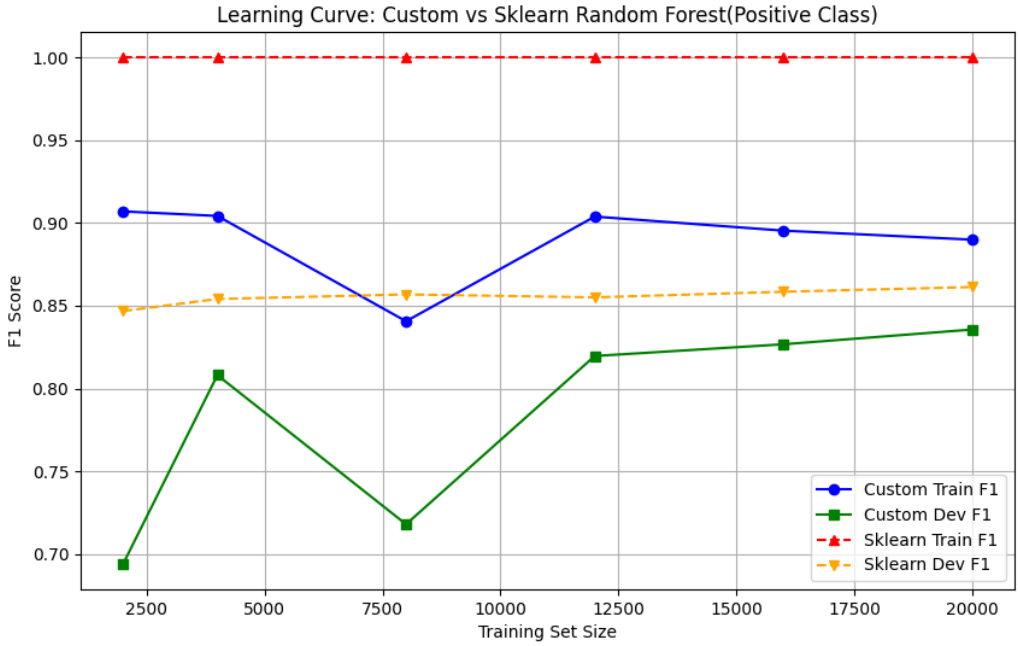
Άρα καλύτερες υπερπαράμετροι οι {'n_estimators': 700, 'm': 5000, 'n_most': 50, 'k_rarest': 50}, με Development Accuracy: 83.84%.

Αποτελέσματα

Ο αλγόριθμος Random Forest ο οποίος υλοποιήσαμε είχε Test Accuracy = 82.93%. Ο έτοιμος αλγόριθμος Random Forest της Scikit-learn είχε Test Accuracy = 85.88%.

Παρακάτω είναι τα διάγραμμα με τις καμπύλες μάθησης στα δεδομένα εκπαίδευσης (training data, όσα έχουν χρησιμοποιηθεί σε κάθε επανάληψη) και ανάπτυξης (το πρώτο είναι του Μέρους Α και το δεύτερο του Μέρους Β). Οι παράμετροι στον αλγόριθμο RandomForestClassifier της Scikit-learn επιλέχθηκαν ώστε να είναι οι ίδιοι με την δική μας υλοποίηση:





Παρακάτω είναι ο πίνακας με τις μετρικές στα δεδομένα αξιολόγησης, όταν χρησιμοποιούνται όλα τα δεδομένα εκπαίδευσης.

Custom Random Forest				Sklearn Random Forest			
	Precision	Recall	F1		Precision	Recall	F1
Positive	0.8204	0.8432	0.8317	Positive	0.8581	0.8598	0.8589
Negative	0.8387	0.8154	0.8269	Negative	0.8595	0.8578	0.8587
Micro-averaged	0.8293	0.8293	0.8293	Micro-averaged	0.8588	0.8588	0.8588
Macro-averaged	0.8296	0.8293	0.8293	Macro-averaged	0.8588	0.8588	0.8588

AdaBoost

Υλοποιήσαμε τον αλγόριθμο AdaBoost, χρησιμοποιώντας δέντρα απόφασης βάθους 1 (decision stumps) ως αδύναμους ταξινομητές. Κατά το στάδιο της εκπαίδευσης, κάθε δέντρο επικεντρώνεται στα παραδείγματα που ταξινομούνται λάθος από τα προηγούμενα δέντρα και τα συνδυάζει με έναν

μηχανισμό βαρών. Στο στάδιο του development/validation και του testing, η τελική πρόβλεψη προκύπτει από τον σταθμισμένο συνδυασμό των αδύναμων ταξινομητών.

Μοντέλο

Ο κώδικας του μοντέλου βρίσκεται στο αρχείο adaboost.py.

Το μοντέλο εκπαιδεύεται μέσω της συνάρτησης adaboost_train (εντός του adaboost.py), η οποία δέχεται ως ορίσματα:

- X: np.ndarray, ένας πίνακας με τα διανύσματα που αναπαριστούν τα παραδείγματα εισόδου.
- y: np.ndarray, ένας πίνακας με τις ετικέτες των features.
- T: Αριθμός των επαναλήψεων (boosting iterations).

Κατά την εκπαίδευση, σε κάθε επανάληψη:

1. Επιλέγεται το βέλτιστο stump (αδύναμο μοντέλο) με το χαμηλότερο σταθμισμένο σφάλμα.
2. Υπολογίζεται η βαρύτητα (alpha) του stump, η οποία καθορίζει τη συνεισφορά του στην τελική απόφαση.
3. Ενημερώνονται τα βάρη των δειγμάτων, ώστε τα δύσκολα παραδείγματα να έχουν μεγαλύτερη σημασία στις επόμενες επαναλήψεις.

Η συνάρτηση επιστρέφει μια λίστα από trained decision stumps.

Η πρόβλεψη των κατηγοριών ενός συνόλου παραδειγμάτων γίνεται μέσω της συνάρτησης adaboost_predict (εντός του adaboost.py). Για κάθε παράδειγμα, συνδυάζει τις προβλέψεις των decision stumps μέσω σταθμισμένης ψήφου.

Train & Development

Ο κώδικας για την εκπαίδευση του μοντέλου βρίσκεται στο αρχείο train_adaboost.py.

Οι συναρτήσεις plot_learning_curve_A και plot_learning_curve_B είναι βοηθητικές και δημιουργούν τα διαγράμματα μάθησης.

Για κάθε συνδυασμό υπερπαραμέτρων, εκπαιδεύουμε:

- Την custom υλοποίηση του AdaBoost.
- Την έτοιμη υλοποίηση του Scikit-learn (με ίδιο αριθμό επαναλήψεων και δέντρα απόφασης βάθους 1) για σύγκριση των αποτελεσμάτων.

Όταν βρεθεί ο καλύτερος συνδυασμός υπερπαραμέτρων (με κριτήριο το development accuracy), αποθηκεύονται:

- Το μοντέλο Custom AdaBoost.
- Το μοντέλο Scikit-learn AdaBoost.
- Το λεξιλόγιο του μοντέλου.
- Τα διαγράμματα μάθησης.

Επιλογή υπερπαραμέτρων:

- T: Δοκιμή από το σύνολο [50,100,150,200,300], οι τιμές 200 και 300 είχαν το υψηλότερο dev_acc αλλά διαλέξαμε την τιμή 200 γιατί με την τιμή 300 το μοντέλο έκανε overfit.
- m_values: Δοκιμή από το σύνολο [1000,2000,3000,4000,5000,7000,10000], οι τιμές εκτός των 5000,7000 είχαν αρκετά χαμηλότερο dev_acc οπότε απορρίφθηκαν. Η καλύτερη ήταν 5000.
- n_most_values: Δοκιμή από το σύνολο [20,50,100], η καλύτερη τιμή ήταν 50.
- k_rarest_values: Δοκιμή από το σύνολο [20,50,100], η καλύτερη τιμή ήταν 50.

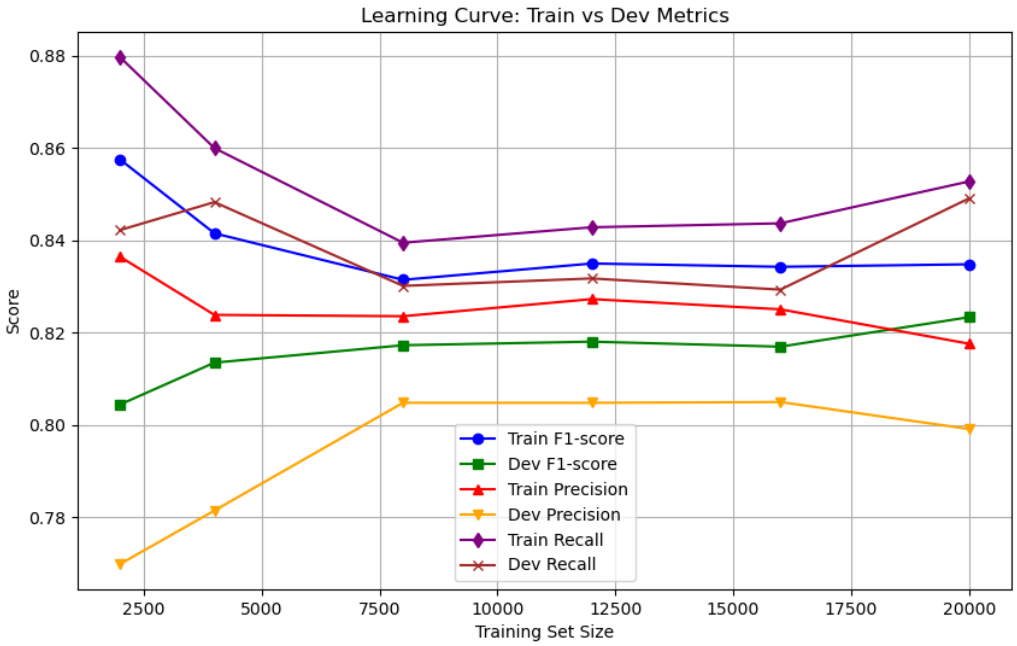
Άρα καλύτερες υπερπαράμετροι οι {'T': 200, 'm': 5000, 'n_most': 50, 'k_rarest': 50}, με Development Accuracy: 82.60% και Test Accuracy: 82.37%.

Αποτελέσματα

Παρακάτω είναι τα διαγράμματα με τις καμπύλες μάθησης στα δεδομένα εκπαίδευσης (training data, όσα έχουν χρησιμοποιηθεί σε κάθε επανάληψη) και ανάπτυξης (development data, πάντα όλα τα development δεδομένα). Οι παράμετροι στον αλγόριθμο AdaboostClassifier της Scikit-learn επιλέχθηκαν ώστε να είναι οι ίδιοι με την δική μας υλοποίηση:

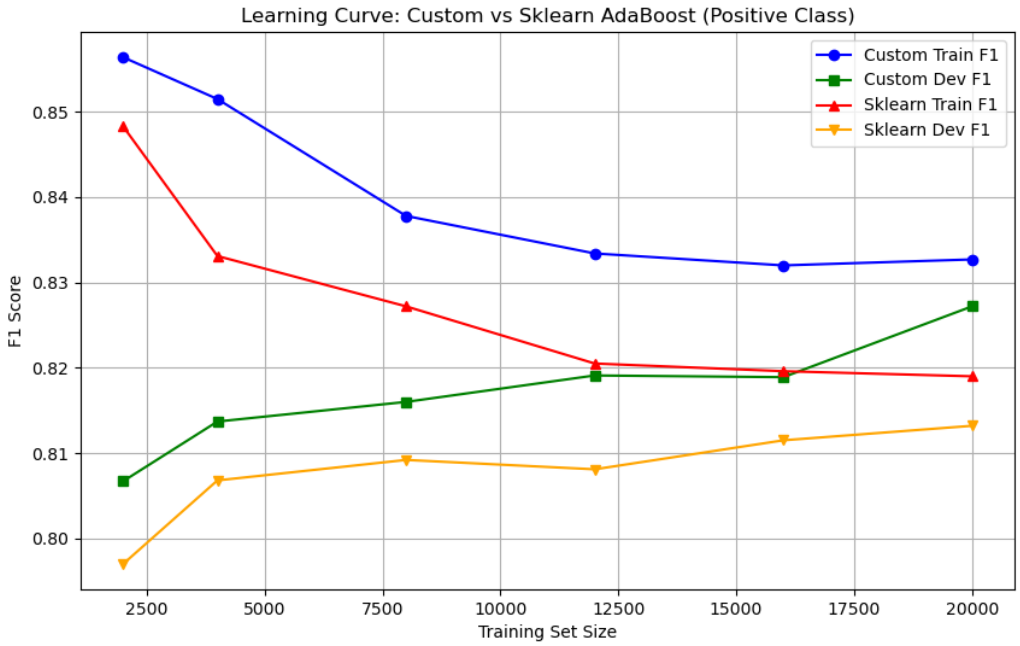
Διάγραμμα Μέρους Α: Custom AdaBoost

Το πρώτο διάγραμμα συγκρίνει precision, recall και F1-score για το training και development set ως συνάρτηση του αριθμού των παραδειγμάτων εκπαίδευσης.



Διάγραμμα Μέρους Β: Custom vs Sklearn AdaBoost

Το δεύτερο διάγραμμα συγκρίνει τις F1-scores του Custom AdaBoost και του Scikit-learn AdaBoost.



Παρακάτω είναι ο πίνακας με τις μετρικές στα δεδομένα αξιολόγησης, όταν χρησιμοποιούνται όλα τα δεδομένα εκπαίδευσης.

Custom Adaboost				Sklearn Adaboost			
	Precision	Recall	F1		Precision	Recall	F1
Positive	0.8098	0.8462	0.8276	Positive	0.7875	0.8428	0.8142
Negative	0.8390	0.8012	0.8197	Negative	0.8309	0.7726	0.8007
Micro-averaged	0.8237	0.8237	0.8237	Micro-averaged	0.8077	0.8077	0.8077
Macro-averaged	0.8244	0.8237	0.8236	Macro-averaged	0.8092	0.8077	0.8075

Stacked Bidirectional RNN (Μέρος Γ)

Υλοποιήσαμε ένα stacked bidirectional RNN με GRU (Gated Recurrent Units) cells για ταξινόμηση κειμένου, χρησιμοποιώντας προεκπαιδευμένα embeddings. Το μοντέλο λαμβάνει ως είσοδο μία σειρά από λέξεις και προβλέπει αν το κείμενο είναι θετικό ή αρνητικό.

Μοντέλο

Ο κώδικας του μοντέλου βρίσκεται στο αρχείο rnnmodel.py.

Το μοντέλο αποτελείται από:

- Embedding Layer: Χρησιμοποιεί προεκπαιδευμένα embeddings για τη μετατροπή των λέξεων σε αριθμητικές αναπαραστάσεις.
- Bidirectional GRU: Δύο μονάδες GRU που διαβάζουν το κείμενο προς τα εμπρός και προς τα πίσω, επιτρέποντας καλύτερη κατανόηση του συμφραζόμενου.
- Global Max Pooling Layer: Εξάγει τις πιο σημαντικές πληροφορίες από την έξοδο του GRU.
- Fully Connected Layer: Χρησιμοποιεί τη μέγιστη ενεργοποίηση από το GRU για να προβλέψει την κατηγορία του κειμένου.

Η εκπαίδευση πραγματοποιείται μέσω της συνάρτησης train_rnnmodel.py, ενώ η πρόβλεψη στις δοκιμαστικές κριτικές γίνεται μέσω της test_rnnmodel.py.

Train & Development

Ο κώδικας για την εκπαίδευση του μοντέλου βρίσκεται στο αρχείο train_rnnmodel.py.

Για κάθε σύνολο υπερπαραμέτρων, το μοντέλο:

- Μετατρέπει τα κείμενα σε ακολουθίες λέξεων, χρησιμοποιώντας ένα λεξικό (vocabulary).
- Χρησιμοποιεί προεκπαιδευμένα word embeddings για την αρχικοποίηση της embedding layer.
- Εκπαιδεύεται με τον Adam optimizer και την CrossEntropy Loss.
- Καταγράφει το train/dev loss για την ανάλυση της σύγκλισης.

Το καλύτερο μοντέλο αποθηκεύεται, και δημιουργούνται καμπύλες μάθησης.

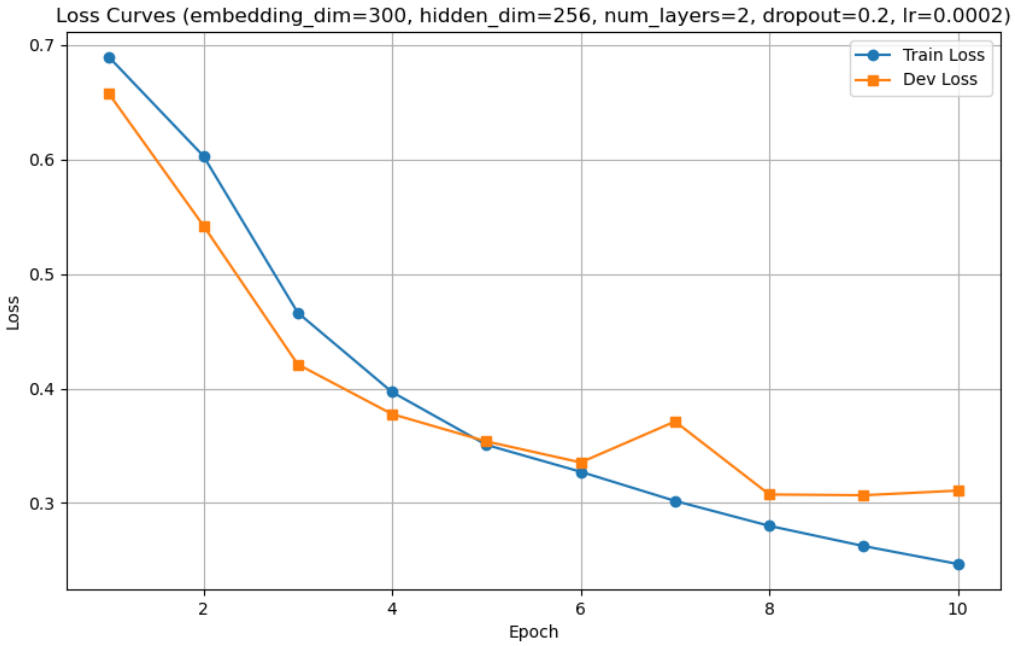
Επιλογή υπερπαραμέτρων:

- embedding_dim: Δοκιμή από το σύνολο [300,400], διαλέξαμε την τιμή 300 γιατί με την τιμή 400 το μοντέλο έκανε overfit.
- hidden_dim: Δοκιμή από το σύνολο [128,256,512], η τιμή 128 είχε αρκετά χαμηλότερο dev_acc οπότε απορρίφθηκε και η τιμή 512 έκανε overfit.
- num_layers: Δοκιμή από το σύνολο [2,3,4], η καλύτερη τιμή ήταν τιμή 2.
- dropout: Δοκιμή από το σύνολο [0.2,0.3,0.5], η καλύτερη τιμή ήταν τιμή 0.2.
- lr: Δοκιμή από το σύνολο [0.0001,0.0002,0.0005,0.001], η τιμή 0.0002 είχε το καλύτερο dev_acc.
- num_epochs: Δοκιμή από το σύνολο [10,15,20]. Η τιμή 10 είχε καλύτερο dev_acc οι τιμές 15 και 20 έκαναν overfit.
- batch_size: Δοκιμή από το σύνολο [64]

Άρα καλύτερες υπερπαράμετροι οι {'embedding_dim': 300, 'hidden_dim': 256, 'num_layers': 2, 'dropout': 0.2, 'lr': 0.0002, 'num_epochs': 10, 'batch_size': 64}, με Development Accuracy: 87.00% και Test Accuracy: 86.91%.

Αποτελέσματα

Το παρακάτω διάγραμμα δείχνει το τις καμπύλες train/dev loss καθώς αυξάνονται οι εποχές εκπαίδευσης.



Παρακάτω παρουσιάζονται οι μετρικές στα δεδομένα αξιολόγησης (test data), όταν χρησιμοποιούνται όλα τα δεδομένα εκπαίδευσης.

	Precision	Recall	F1
Positive	0.8686	0.8698	0.8692
Negative	0.8697	0.8684	0.8690
Micro-averaged	0.8691	0.8691	0.8691
Macro-averaged	0.8691	0.8691	0.8691