University of Athens
Department of Informatics and Telecommunications

Deep Learning for NLP

Student name: *Βασιλική Τσαντήλα*

Course: *Artificial Intelligence II*
Semester: *Spring Semester 2025*

Περιεχόμενα

1. Abstract

Έχοντας ως δομικό στοιχείο ένα pretrained model, στόχος μας είναι η ανάπτυξη ενός sentiment classifier για ταξινόμηση συναισθήματος σε δοσμένο dataset από tweets.

2. Data processing and analysis

Ο λόγος που στα cells των sections 1 και 2 των notebooks, δεν φαίνονται τα αντίστοιχα outputs, είναι διότι τα συγκεκριμένα cells εκτελέστηκαν πολλές φορές με τις διαφορετικές τιμές που υποδεικνύονται στο παρόν report.

2.1 Pre-processing

Εκτυπώνουμε μερικά από τα δεδομένα μας:

```
0       @whoisralphie dude I'm so bummed ur leaving!
1       oh my god, a severed foot was found in a wheely bin in cobham!!! where
        they found is literally minutes from my house! feel sick now!
2       I end up "dog dialing" sometimes. What's dog dialing, u ask? My dogs
        will walk across my phone & end up calling someone.
3       @_rachelx meeeee toooooo!
4       I was hoping I could stay home and work today, but looks like I have to
        make another trip into town
148383 just love the jonas brothers its too bad i will never get to see them
        tears
148384 another day gone by....time is moving so fast...
148385 fuck college, i'm just gonna marry rich. : fuck college, i'm just gonna
        marry rich.
148386 ZOMGZ NEW SONG FTW. remember that night. <3
148387 http://twitpic.com/7mwrd - Arby's took down their Roastburger coupon.
        But i found the image in my browser cache...so...cheap lunch ...
```

Αναλύοντας τα δεδομένα μάς σκεφτόμαστε, ότι θα είναι χρήσιμο να μετατρέπουμε κάθε κεφαλαίο γράμμα σε πεζό, καθώς θέλουμε οι λέξεις π.χ. 'Good' και 'good' το μοντέλο να τις καταλάβει ως την ίδια λέξη (my_lower).
Επιπλέον, βλέπουμε πως οι προτάσεις μας αποτελούνται από stopwords, λέξεις που δεν προσδίδουν κάποια χρησιμότητα στο μοντέλο στην προσπάθειά του να ερμηνεύσει αν η πρόταση έχει θετικό ή αρνητικό πρόσημο (συναισθηματικά) (my_stopword).
Σκεφτόμαστε, πως μπορούμε να δοκιμάσουμε να αφαιρέσουμε τα σημεία στίξης, και να δούμε αν αυτό θα επιφέρει κάποια βελτίωση ή όχι στο μοντέλο (my_unpunct).
Συνειδητοποιούμε ότι το ίδιο ρήμα εμφανίζεται στο dataset σε διαφορετικές μορφές. Έτσι, σκεφτόμαστε ότι θα είναι χρήσιμο κάθε ρήμα να το μετατρέπουμε στη βασική του μορφή (my_lemmatize).
Αντιλαμβανόμαστε ότι στην εν λόγω άσκηση δεν υπάρχει κάποια απαίτηση για ανωνυμία και προστασία των προσωπικών δεδομένων, επομένως επιλέγουμε να αφήσουμε τυχόν usernames, e-mails κλπ ως έχουν, χωρίς κάποια επιπλέον επεξεργασία.
Σημειώνουμε, ότι η παραπάνω επεξεργασία εφαρμόζεται αφού έχει γίνει πρώτα tokenized το κείμενο (my_split).

Εφαρμόζοντας τις παραπάνω συναρτήσεις προ-επεξεργασίας, παρατηρούμε ότι πλέον τα δεδομένα μας έχουν την εξής μορφή:

```
0       whoisralphie dude  im bummed ur leaving
1       oh god severed foot foun wheely bin cobham found literally minute house
        feel sick now
2       end quotdog dialingquot sumtimes whats dog dialing u askmy dog walk
        across
        phone amp end calling someone aka  quotdog dialingquot
3       rachelx meeeee toooooo
4       hoping could stay home work today look like make another trip town
148383 love jonas brother  tooo bad wil never get see  tear
148384 another day gone bytime moving fast
148385 fuck college im gonna marry rich  fuck college im gonna marry rich
        httpbitlymgic4
148386 zomgz new song ftw  remember night lt3
148387 httptwitpiccom7mwrd  arbys took roastburger coupon     found image
        browser cachesocheap lunch
```

2.2 Data partitioning for train, test and validation

Τα δεδομένα μας ήταν ήδη από την εκφώνηση χωρισμένα σε train, validation και test set, συνεπώς δεν υπέστη κάποιο διαφορετικό διαχωρισμό από εμάς.

2.3 Tokenization

Για το BERT χρησιμοποιούμε τον transformers.BertTokenizer, ενώ για το DistilBert τον transformers.DistilBertTokenizer.
Παρατηρούμε ότι οι προτάσεις μας διαφέρουν σε μήκος. Έτσι, ορίζουμε οι προτάσεις μας να έχουν μήκος όσο το μήκος της μεγαλύτερης πρότασης που βρήκαμε συνολικά στα training, validation και test set, περιλαμβανομένων των tokens [CLS] και [SEP]. Το padding για τις προτάσεις που έχουν μικρότερο μήκος από το μέγιστο μήκος, γίνεται από τον tokenizer (tokenizer.encode_plus).

3. BERT

3.1 Experiments

Σημαντική παρατήρηση: Όπως θα φανεί και στην παρακάτω ανάλυση εκτελέστηκαν πολλά πειράματα με στόχο την εύρεση του βέλτιστου δυνατού τελικού μοντέλου. Ωστόσο, δεδομένου ότι οι ώρες στο kaggle με χρήση GPU είναι περιορισμένες και για τα 2 notebooks, δεν ήταν εφικτό να εκτελεστεί κάθε επιθυμητό πείραμα. Υπάρχουν σημεία στο παρακάτω report στα οποία αναφέρω ποια επιπλέον πειράματα θα εκτελούσα και πόσα θα ήταν αυτά, αν η διαθεσιμότητα σε GPU ήταν μεγαλύτερη.

3.1.1 Preprocessing

Χρησιμοποιούμε το BertForSequenceClassification, ένα προεκπαιδευμένο μοντέλο BERT με 12 layers, το οποίο επεκτείνεται με ένα πρόσθετο linear layer ταξινόμησης on top, λειτουργώντας ως sentence classifier. Το μοντέλο που επιλέγουμε χρησιμοποιεί uncased λεξιλόγιο (bert-base-uncased).

Θα ξεκινήσουμε από το μοντέλο που φαίνεται παρακάτω, και θα ελέγξουμε ποιο είδους preprocessing βοηθά το μοντέλο μας να πετύχει το καλύτερο accuracy στο validation set. Το συγκεκριμένο πείραμα θα το ξανά τρέξουμε και στο τελικό μας μοντέλο.

Batch size	32
Learning rate	1e-5
Epochs	1
Optimizer	AdamW
eps	1e-8
Tokenizer	do_lower_case=True
Learning Scheduler	unavailable for now
L2 Norm Clipping	unavailable for now

Preprocessing	Accuracy in val_set
disabled	0.849989892
enabled all	0.793530997
disabled stopword	0.839353100
disabled lemmatize	0.795111186
disabled stopword, lemmatize	0.839356469

Table 1: Testing preprocessing in initial BERT model

Ο BertTokenizer που χρησιμοποιεί το μοντέλο μας έχει ρυθμιστεί με την παράμετρο do_lower_case=True. Συνεπώς, σε κάθε περίπτωση το κατ΄ ελάχιστον preprocessing που λαμβάνουν οι προτάσεις μας (ακόμα και αν εμείς δεν κάνουμε preprocessing - π.χ. all preprocessing disabled) είναι να γίνονται όλες uncased από τον tokenizer.

Στο παραπάνω πινακάκι δεν έχει δοκιμαστεί η περίπτωση: disabled stopword, lemmatize, unpunct. Ο λόγος είναι επειδή η περίπτωση αυτή ταυτίζεται με το να είναι ενεργοποιημένη μόνο η συνάρτηση my_lower του preproccesing, η οποία όπως εξηγήσαμε και προηγουμένως έχει καλυφθεί από την περίπτωση: all preprocessing disabled.

Αυτό που βλέπουμε στο πινακάκι αποτελεσμάτων είναι ότι με αρκετά σημαντική διαφορά πετυχαίνουμε καλύτερο accuracy στο validation set όταν το μοναδικό preprocessing που συμβαίνει στις προτάσεις μας, είναι εκείνο του tokenizer (δηλαδή, όταν όλες οι προτάσεις γίνονται uncased).

Στην περαιτέρω ανάλυση μας, το μοναδικό preprocessing που εφαρμόζουμε στις προτάσεις μας είναι εκείνο του tokenizer (lowercase), μιας που μάς έδωσε τα καλύτερα αποτελέσματα. Όπως αναφέραμε και προηγουμένως, το συγκεκριμένο πείραμα θα εκτελεστεί εκ νέου και στο τελικό μοντέλο.

3.1.2 Hyper-parameter tuning

Στο section "A.3 Fine-tuning Procedure" του αντίστοιχου paper στο οποίο παρουσιάζεται το BERT (όπως και στις διαφάνειες της θεωρίας), επισημαίνεται πως αν θέλουμε να κάνουμε fine-tune το μοντέλο μας, τότε οι υπεραπαράμετροι προς πειραματισμό είναι οι εξής: batch size, learning rate και πλήθος εποχών. Ειδικότερα, το εύρος τιμών για την κάθε υπερπαράμετρο συνιστάται να είναι:

Batch size	16, 32
Learning rate (Adam)	5e-5, 3e-5, 2e-5
Epochs	2, 3, 4

Με γνώμονα τις οδηγίες του paper, δοκιμάζουμε ποιες από τις προτεινόμενες τιμές μάς

δίνουν τα καλύτερα αποτελέσματα για το δικό μας task.
Συνολικά, εκτελούμε 6 trials. Κάθε trial τρέχει για max 5 εποχές.
Ο optimizer που χρησιμοποιούμε είναι ο AdamW με eps = 1e-8.
Οι τιμές learning rate και batch size που δοκιμάζουμε είναι αυτές που υποδεικνύονται στο paper.
Υλοποιούμε early stopping με patience = 2 και μετρική προς παρακολούθηση το validation loss. Έτσι, αν έχουν περάσει 2 εποχές και *δεν* έχει σημειωθεί μείωση του validation loss, τότε η διαδικασία του training διακόπτεται και δεν προχωράμε σε επόμενη εποχή. Ο λόγος που υλοποιούμε early stopping είναι προκειμένου να αποφύγουμε το μοντέλο μας να κάνει overfit. Ο λόγος που επιλέγουμε το validation loss, αντί για το validation accuracy ως τη μετρική που θα καθορίσει πότε θα σταματήσουμε το training, είναι διότι ακόμα κι αν το μοντέλο προβλέπει σωστά μια κλάση, αλλά με μικρότερη βεβαιότητα, το validation loss θα αυξηθεί, ενώ το accuracy θα παραμείνει το ίδιο. Το validation loss μετράει την "απόσταση" των προβλέψεων από τις πραγματικές τιμές, ενώ το accuracy αγνοεί αυτές τις αποκλίσεις, εφόσον η πρόβλεψη ήταν σωστή.
Χρησιμοποιούμε learning rate scheduler προκειμένου να γίνεται δυναμική τροποποίηση του learning rate.
Εφαρμόζουμε L2 norm clipping, επιβάλλοντας τον υπολογισμό της L2 (ευκλείδειας) νόρμας όλων των gradients και αν η τιμή της νόρμας ξεπερνά το 1.0, τότε οι gradients μειώνοται προκειμένου η (συνολική) νέα νόρμα των gradients να ισούται με 1.0. Με αυτόν τον τρόπο αποφεύγουμε το πρόβλημα των exploding gradients (gradients grow excessively large).
Από άποψη preprocessing και τα 6 trials εκτελούνται έχοντας ενεργοποιημένο μόνο το lowercase του tokenizer.
Ακολουθούν τα αποτελέσματα των trials:

- Trial #1: 'epochs': 5, 'batch_size': 16, 'learning_rate': 5e-05

  Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.411072250	0.405095859	0.830070755
2	0.318348914	0.432781586	0.840597484
3	0.231724957	0.447166949	0.842177673

Table 2: Fine-Tuning BERT: Trial #1

- Trial #2: 'epochs': 5, 'batch_size': 32, 'learning_rate': 5e-05

  Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.385360100	0.364474125	0.845791779
2	0.270634891	0.376754547	0.846991240
3	0.158886180	0.487207615	0.844609164

Table 3: Fine-Tuning BERT: Trial #2

- Trial #3: 'epochs': 5, 'batch_size': 16, 'learning_rate': 3e-05

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.386445511	0.360221588	0.847287736
2	0.276098420	0.377852530	0.848231132
3	0.185279274	0.501314402	0.849433962

Table 4: Fine-Tuning BERT: Trial #3

Early stopping at epoch 3.

- Trial #4: 'epochs': 5, 'batch_size': 32, 'learning_rate': 3e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.376666952	0.347080263	0.852156334
2	0.258102500	0.359312901	0.855010108
3	0.156824705	0.469050822	0.852223720

Table 5: Fine-Tuning BERT: Trial #4

- Trial #5: 'epochs': 5, 'batch_size': 16, 'learning_rate': 2e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.380077882	0.358100383	0.851509434
2	0.273976968	0.369607056	0.853349057
3	0.191326181	0.500153033	0.850676101

Table 6: Fine-Tuning BERT: Trial #5

- Trial #6: 'epochs': 5, 'batch_size': 32, 'learning_rate': 2e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.376171492	0.347145849	0.851994609
2	0.270955483	0.356902642	0.854349730
3	0.188846546	0.418721393	0.853264825

Table 7: Fine-Tuning BERT: Trial #6

Συμπεράσματα:
Παρατηρούμε ότι κάθε trial κάνει early stopping στην εποχή 3. Άρα, μέχρι την εποχή 2 έχει ήδη σημειωθεί η καλύτερη απόδοση για το κάθε trial, και το περαιτέρω training επιφέρει overfit στο μοντέλο.
Παρατηρούμε ότι για κάθε ζεύγος trials με ίδιο learning rate, σημειώνεται καλύτερη απόδοση όταν το batch size είναι 32 από όταν το batch size είναι 16.
Παρατηρούμε ότι το trial #4 είναι αυτό που πετυχαίνει το χαμηλότερο validation loss και

το υψηλότερο validation accuracy με τιμές 0.347080263 και 0.855010108, αντίστοιχα. Εφόσον το trial #4 είναι με batch size = 32, σκεφτόμαστε αν η αύξηση του batch size σε 64, ενδεχομένως να επιφέρει ακόμα καλύτερα αποτελέσματα για το μοντέλο μας, κρατώντας το learning rate σταθερό (3e-5). Δοκιμάζουμε, λοιπόν, ένα ακόμα πείραμα:

- Trial #7: 'epochs': 5, 'batch_size': 64, 'learning_rate': 3e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.376721798	0.345662177	0.852112961
2	0.268847120	0.353489140	0.851318412
3	0.174643782	0.424845128	0.851092841

Table 8: Fine-Tuning BERT: Trial #7

Παρατηρούμε ότι δεν υπήρξε βελτίωση του μοντέλου με τη τροποποίηση του batch_size σε 64. Κρατάμε, λοιπόν, ως καλύτερες τις τιμές των υπερπαραμέτρων του trial #4: 'epochs': 2, 'batch_size': 32, 'learning_rate': 3e-05

Τέλος, πειραματιζόμαστε με το weight_decay του AdamW optimizer. Η default τιμή του weight_decay για τον AdamW είναι 0.01. Εμείς δοκιμάζουμε τρεις επιπλέον τιμές: 0.0, 0.05 και 0.1. Ο λόγος που το κάνουμε αυτό είναι για να δούμε κατά πόσο συμβάλλει αυτή η παράμετρος στην αποφυγή overfitting. Ιδανικά, το τρέχον πείραμα θα το τρέχαμε για κάθε trial από τα 7 που υλοποιήθηκαν προηγουμένως, δεδομένου όμως ότι οι διαθέσιμες ώρες στο Kaggle με χρήση GPU είναι περιορισμένες και για τα 2 notebooks, πραγματοποιούμε το τρέχον πείραμα μόνο για το trial που μάς έδωσε τα καλύτερα αποτελέσματα προηγουμένως (δηλαδή, για το trial #4).

- Trial #8: 'epochs': 5, 'batch_size': 32, 'learning_rate': 3e-05, 'weight_decay': 0.0

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.378395241	0.349439146	0.850438005
2	0.265674248	0.355806807	0.852277628
3	0.163855318	0.451209042	0.849278976

Table 9: Fine-Tuning BERT: Trial #8

- Trial #9: 'epochs': 5, 'batch_size': 32, 'learning_rate': 3e-05, 'weight_decay': 0.05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.378814357	0.349379238	0.850886119
2	0.267662970	0.359125809	0.852065364
3	0.166211080	0.467972842	0.850316712

Table 10: Fine-Tuning BERT: Trial #9

- Trial #10: 'epochs': 5, 'batch_size': 32, 'learning_rate': 3e-05, 'weight_decay': 0.1

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.378495330	0.351607241	0.850080863
2	0.267694547	0.355567883	0.852018194
3	0.166245858	0.446097514	0.850411051

Table 11: Fine-Tuning BERT: Trial #10

Συμπεράσματα:
Παρατηρούμε ότι η διαφοροποίηση της τιμής του weight decay επιφέρει πολύ μικρές διαφορές στο validation loss και στο validation accuracy. Ενδεχομένως, να μην μπορούμε να δούμε πόσο πραγματικά η παράμετρος αυτή επηρεάζει την απόδοση του μοντέλου μας, δεδομένου ότι υλοποιούμε early stopping με μικρό patience και έτσι συνολικά το μοντέλο μας τρέχει για μικρό πλήθος εποχών. Σε αυτές τις λίγες εποχές που τρέχει το μοντέλο μας, παρατηρούμε ότι το καλύτερο αποτέλεσμα επιτυγχάνεται όταν η παράμετρος weight decay έχει την default τιμή ίση με 0.01.

Αν οι διαθέσιμες ώρες στο kaggle ήταν περισσότερες με χρήση GPU, τότε θα εκτελούσαμε τα πειράματα του section 3.1.2 Hyper-parameter tuning με ενεργοποιημένες κάθε συνάρτηση του preprocessing. Εν συνεχεία, θα επιλέγαμε ως τελικό μοντέλο το μοντέλο που έδωσε τα καλύτερα αποτελέσματα από τα 10 + 10 trials (χωρίς preprocessing vs με preprocessing). Επειδή, δεν υπάρχει αυτή η ευχέρεια, θεωρούμε ως καλύτερο μοντέλο αυτό που βρήκαμε στην μέχρι τώρα ανάλυσή μας το οποίο είναι το εξής: 'epochs': 2, 'batch size': 32, 'learning rate': 3e-05, 'weight decay': 0.01.

3.1.3 Final model

Όπως αναφέραμε και στην υποενότητα 3.1.1 Bert/Experiments/Preprocessing, τώρα που έχουμε το τελικό μας μοντέλο, ελέγχουμε αν πράγματι παίρνουμε την καλύτερη απόδοση όταν το μοναδικό preprocessing που εφαρμόζουμε είναι εκείνο του tokenizer. Επαναλαμβάνουμε, δηλαδή, το πείραμα της υποενότητας 3.1.2, υλοποιώντας όμως early stopping, learning rate scheduler, L2 norm clipping και εφαρμόζοντας weight decay ίσο με 0.01. Ορίζουμε ο αριθμός των εποχών σε κάθε trial να είναι 5.

Preprocessing	Accuracy in val_set
disabled	0.855010108 in epoch = 2
enabled all	0.797641509 in epoch = 2
disabled stopword	0.840060647 in epoch = 1
disabled lemmatize	0.798463612 in epoch = 2
disabled stopword, lemmatize	0.840370620 in epoch = 1

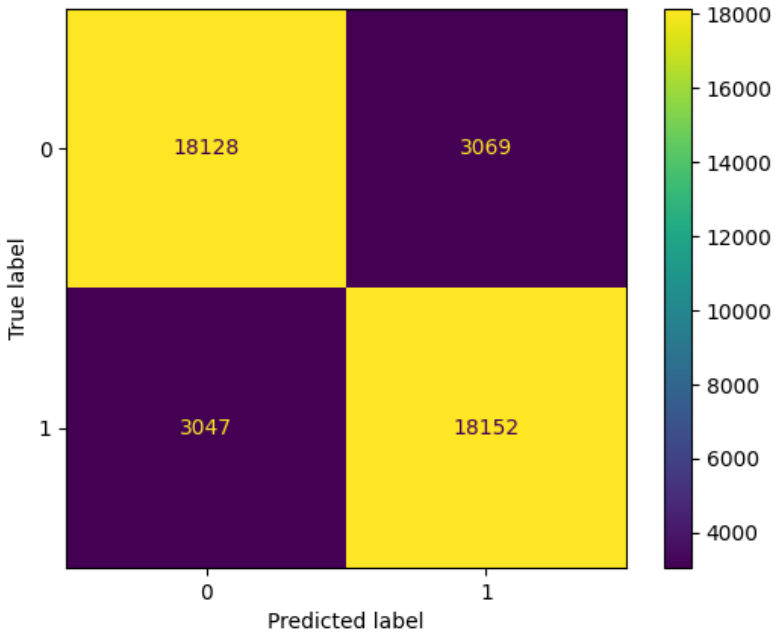
Table 12: Testing preprocessing in final model

Παρατηρούμε πως πράγματι το μοντέλο μας έχει την καλύτερη απόδοση όταν το μοναδικό preprocessing που εφαρμόζουμε είναι εκείνο του tokenizer. Καταλήγουμε, δηλαδή, ότι το καλύτερο μοντέλο μας έχει ως εξής:

Preprocessing	tokenizer's (do_lower_case=True)
Batch size	32
Learning rate	3e-5
Epochs	2
Optimizer	AdamW
Weight decay	0.01
eps	1e-8
Learning Scheduler	yes
L2 Norm Clipping	yes

3.2 Final model evaluation

3.2.1 Confusion matrix



Σχήμα 1: BERT model: Confusion matrix

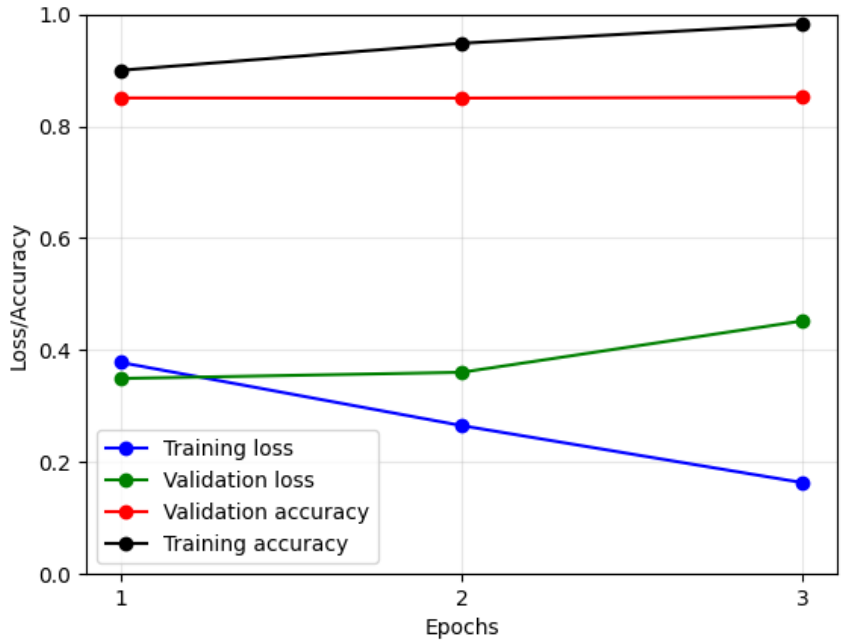
Class	Precision	Recall	F1-score	Support
0	0.86	0.86	0.86	21197
1	0.86	0.86	0.86	21199

Table 13: BERT model: Classification report

Το μοντέλο μας πετυχαίνει 86% accuracy. Αποτελεί, δηλαδή, βελτίωση του αρχικού μοντέλου κατά 1%. Από το classification report βλέπουμε ότι κάθε μετρική έχει την ίδια τιμή και στις δύο κλάσεις. Αυτό δείχνει ότι το μοντέλο παρουσιάζει σταθερή απόδοση και στις δύο κλάσεις χωρίς να ευνοεί μια κλάση έναντι της άλλης. Από την στήλη 'support' του classification report καταλαβαίνουμε ότι το dataset είναι balanced (21197 δείγματα με θετικό label vs 21199 δείγματα με αρνητικό label). Τέλος, από τα αποτελέσματα στον confusion

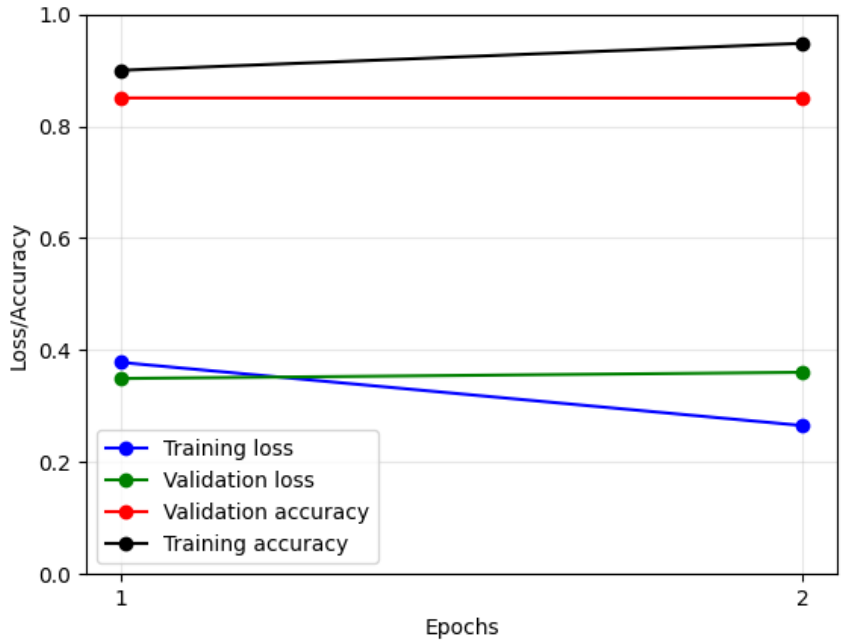
matrix βλέπουμε ότι το πλήθος των σφαλμάτων του μοντέλου είναι: 3.069 false positives και 3.047 false negatives, γεγονός που επιβαβαιώνει την ισορροπημένη συμπεριφορά του μοντέλου, καθώς ο αριθμός των false positives και των false negatives είναι σχεδόν ίδιος.

3.2.2 Learning curve



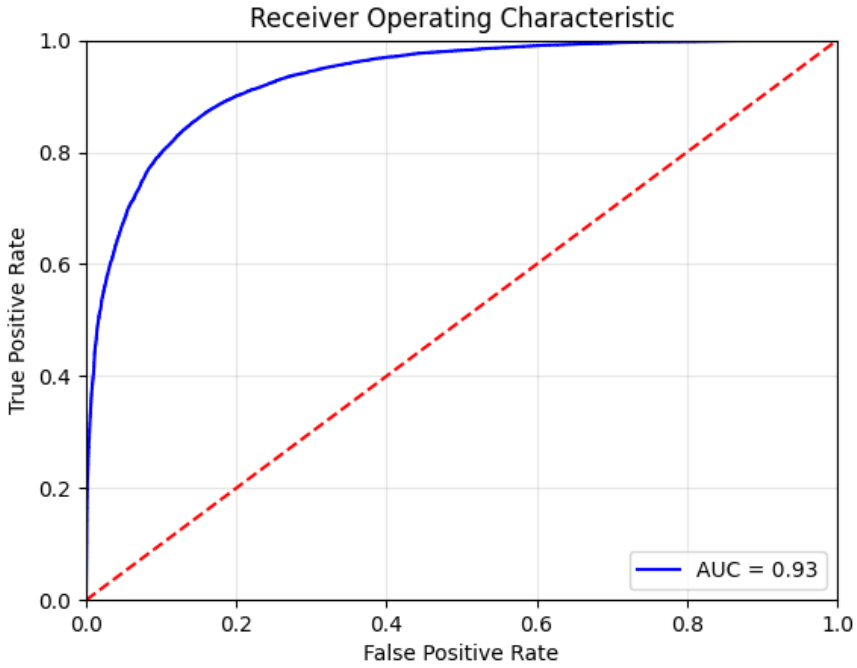
Σχήμα 2: BERT model (3 epochs): Learning curve

Μελετώντας τη καμπύλη μάθησης του μοντέλου βλέπουμε ότι δεν υπάρχει σημαντική βελτίωση του validation accuracy με το πέρασμα των εποχών, ενώ για το training accuracy υπάρχει σημαντική αύξηση. Ταυτοχρόνως, βλέπουμε ότι από την εποχή 2 και μετά υπάρχει εμφανής απόκλιση του training loss vs validation loss, γεγονός που υποδεικνύει ότι το μοντέλο μας από την εποχή 2 και μετά κάνει overfit. Μέσω της καμπύλης μάθησης του μοντέλου αιτιολογούμε το early stopping που συμβαίνει στην εποχή 3 (με patience = 2), δίνοντας μας ένα τελικό (βέλτιστο) μοντέλο μετά από μόλις 2 εποχές (Σχήμα 3).
Σημειώνουμε ότι το τελικό μοντέλο του σχήματος 3 πετυχαίνει accuracy στο test set ίσο με 0.85692.



Σχήμα 3: BERT **final** model (2 epochs): Learning curve

3.2.3 ROC curve



Σχήμα 4: BERT model: ROC curve

Αναφορικά με τη Reciever Operating Characteristic Curve (ROC) του τελικού μοντέλου, παρατηρούμε ότι η Area Under the Curve (AUC) είναι εξαιρετικά μεγαλύτερη από

0.5 (τιμή που αποδίδεται σε ένα μοντέλο που τυχαία δίνει τιμές στα predictions του). Η τιμή της AUC ισούται με 0.93, αποτυπώντας την ικανότητα του μοντέλου να διακρίνει με υψηλή αξιοπιστία τις δύο κλάσεις/labels.

4. DistilBERT

4.1 Experiments

4.1.1 Preprocessing

Χρησιμοποιούμε το DistilBertForSequenceClassification, ένα μοντέλο με 6 layers, το οποίο επεκτείνεται με ένα πρόσθετο linear layer ταξινόμησης on top, λειτουργώντας ως sentence classifier. Το μοντέλο που επιλέγουμε χρησιμοποιεί uncased λεξιλόγιο (distilbert-base-uncased).

Θα ξεκινήσουμε από το μοντέλο που φαίνεται παρακάτω (όπως κάναμε και στο BERT), και θα ελέγξουμε ποιο είδους preprocessing βοηθά το μοντέλο μας να πετύχει το καλύτερο accuracy στο validation set.

Batch size	32
Learning rate	1e-5
Epochs	1
Optimizer	AdamW
eps	1e-8
Tokenizer	do_lower_case=True
Learning Scheduler	unavailable for now
L2 Norm Clipping	unavailable for now

Preprocessing	Accuracy in val_set
disabled	0.83805593
enabled all	0.77440027
disabled stopword	0.827152965
disabled lemmatize	0.776688005
disabled stopword, lemmatize	0.827631402

Table 14: Testing preprocessing in initial DistilBERT model

Παρόμοια με το BERT, το κατ΄ ελάχιστον preprocessing που λαμβάνουν οι προτάσεις μας (ακόμα και αν εμείς δεν κάνουμε preprocessing) είναι να γίνονται όλες uncased από τον tokenizer, δεδομένου ότι ο tokenizer έχει ρυθμιστεί με την παράμετρο do_lower_case=True.

Παρατηρούμε ότι και σε αυτό το μοντέλο, παίρνουμε τα καλύτερα αποτελέσματα όταν το μοναδικό preprocessing που συμβαίνει στις προτάσεις μας, είναι εκείνο του tokenizer. Συνεπώς, και εδώ, στην περαιτέρω ανάλυση μας, το μοναδικό preprocessing που εφαρμόζουμε στις προτάσεις μας είναι εκείνο του tokenizer (lowercase).

4.1.2 Hyper-parameter tuning

Όπως και στο BERT, θα ξεκινήσουμε με τον πειραματισμό των υπερπαραμέτρων που αναφέρονται στο section "A.3 Fine-tuning Procedure" του paper. Και εδώ, τρέχουμε κάθε trial για max 5 εποχές. Χρησιμοποιούμε τον AdamW optimizer. Υλοποιούμε learning rate scheduler, L2 norm clipping και early stopping με patience = 2 και μετρική προς παρακολούθηση το validation loss.
Ακολουθούν τα αποτελέσματα των trials:

- Trial #1: 'epochs': 5, 'batch_size': 16, 'learning_rate': 5e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.407306322	0.368868234	0.838364780
2	0.303505121	0.403047007	0.839968553
3	0.206039291	0.489474866	0.836171384

Table 15: Fine-Tuning DistilBERT: Trial #1

- Trial #2: 'epochs': 5, 'batch_size': 32, 'learning_rate': 5e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.397995786	0.362542723	0.839626011
2	0.288020201	0.385987796	0.842574124
3	0.180953263	0.452953710	0.839696765

Table 16: Fine-Tuning DistilBERT: Trial #2

- Trial #3: 'epochs': 5, 'batch_size': 16, 'learning_rate': 3e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.396477470	0.358809858	0.841933962
2	0.292797697	0.385911766	0.844166667
3	0.203056780	0.461835721	0.839701258

Table 17: Fine-Tuning DistilBERT: Trial #3

- Trial #4: 'epochs': 5, 'batch_size': 32, 'learning_rate': 3e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.393353028	0.364144790	0.839188005
2	0.292460105	0.387919215	0.846115229
3	0.201053522	0.427978778	0.843830863

Table 18: Fine-Tuning DistilBERT: Trial #4

- Trial #5: 'epochs': 5, 'batch_size': 16, 'learning_rate': 2e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.393568998	0.356946074	0.843655660
2	0.297795472	0.386350067	0.846540881
3	0.218000199	0.458019513	0.843687107

Table 19: Fine-Tuning DistilBERT: Trial #5

- Trial #6: 'epochs': 5, 'batch_size': 32, 'learning_rate': 2e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.393879825	0.360576120	0.841051213
2	0.304537513	0.373325364	0.848547844
3	0.230524363	0.400020552	0.842934636

Table 20: Fine-Tuning DistilBERT: Trial #6

Συμπεράσματα:
Παρατηρούμε ότι (όπως και στο BERT) κάθε trial κάνει early stopping στην εποχή 3. Παρατηρούμε ότι το trial #5 είναι αυτό που πετυχαίνει το χαμηλότερο validation loss, ενώ το trial #6 είναι αυτό που πετυχαίνει το υψηλότερο validation accuracy, με τιμές 0.356946074 και 0.848547844, αντίστοιχα. Αν και εδώ, σε σύγκριση με το BERT δεν ισχύει πάντα ότι τα trials με batch size = 16 δίνουν καλύτερα αποτελέσματα από τα trials με batch size = 32, δοκιμάζουμε τις παραμέτρους του trial #6 (trial με καλύτερο validation accuracy), αλλά με batch size = 64 αυτή τη φορά.

- Trial #7: 'epochs': 5, 'batch_size': 64, 'learning_rate': 2e-05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.396324298	0.358115208	0.841743428
2	0.313319322	0.359398697	0.847682342
3	0.251102117	0.371464211	0.846268315

Table 21: Fine-Tuning DistilBERT: Trial #7

Παρατηρούμε ότι δεν υπήρξε βελτίωση του μοντέλου με τη τροποποίηση του batch_size σε 64. Κρατάμε, λοιπόν, ως καλύτερες τις τιμές των υπερπαραμέτρων του trial #6:
'epochs': 2, 'batch_size': 32, 'learning_rate': 2e-05
Τέλος, πειραματιζόμαστε με το weight_decay του AdamW optimizer. Έχοντας δοκιμάσει τη default τιμή 0.01, δοκιμάζουμε επιπλέον άλλες τρεις τιμές, και βλέπουμε κατά πόσο επηράζει αυτή η παραμέτρος στην αποφυγή overfitting του μοντέλου μας. Οι δοκιμές αυτές γίνονται επί των παραμέτρων του trial #6.

- Trial #8: 'epochs': 5, 'batch_size': 32, 'learning_rate': 2e-05, 'weight_decay': 0.0

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.393897867	0.360699711	0.840316712
2	0.304476984	0.373525813	0.848028976
3	0.230426220	0.400666643	0.843359164

Table 22: Fine-Tuning DistilBERT: Trial #8

- Trial #9: 'epochs': 5, 'batch_size': 32, 'learning_rate': 2e-05, 'weight_decay': 0.05

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.393919067	0.361103667	0.839962938
2	0.304819420	0.372764546	0.848382749
3	0.231109988	0.400528941	0.842604447

Table 23: Fine-Tuning DistilBERT: Trial #9

- Trial #10: 'epochs': 5, 'batch_size': 32, 'learning_rate': 2e-05, 'weight_decay': 0.1

Early stopping at epoch 3.

epoch	Training Loss	Validation Loss	Validation Accuracy
1	0.394032903	0.360083106	0.840458221
2	0.305246701	0.372770582	0.847769542
3	0.231856285	0.399886359	0.843429919

Table 24: Fine-Tuning DistilBERT: Trial #10

Συμπεράσματα:
Παρατηρούμε ότι η διαφοροποίηση της τιμής του weight decay επιφέρει πολύ μικρές διαφορές στο validation loss και στο validation accuracy. Ενδεχομένως, να μην μπορούμε να δούμε πόσο πραγματικά η παράμετρος αυτή επηρεάζει την απόδοση του μοντέλου μας, δεδομένου ότι υλοποιούμε early stopping με μικρό patience και έτσι συνολικά το μοντέλο μας τρέχει για μικρό πλήθος εποχών. Σε αυτές τις λίγες εποχές που τρέχει το μοντέλο μας, παρατηρούμε ότι το καλύτερο αποτέλεσμα επιτυγχάνεται όταν η παράμετρος weight decay έχει την default τιμή ίση με 0.01.

Όπως και στο BERT ιδανικά θα εκτελούσαμε όσα πειράματα έχουμε πραγματοποιήσει ως στιγμής με ενεργοποιημένο το preprocessing, όμως, αυτή τη φορά. Εφόσον δεν έχουμε αυτά τα επιπλέον αποτελέσματα, περιοριζόμαστε στον ορισμό του καλύτερου μοντέλου από όσα αποτελέσματα έχουμε ως στιγμής. Έτσι, βρίσκουμε ότι το trial #6 μάς δίνει το καλύτερο μοντέλο: 'epochs': 2, 'batch size': 32, 'learning rate': 2e-05, 'weight decay': 0.01.

4.1.3 Final model

Τώρα που έχουμε το τελικό μας μοντέλο, ελέγχουμε αν πράγματι παίρνουμε την καλύτερη απόδοση όταν το μοναδικό preprocessing που εφαρμόζουμε είναι εκείνο του tokenizer. Στο παρακάτω πείραμα μας, υλοποιούμε early stopping, learning rate scheduler, L2 norm

clipping και το weight decay που εφαρμόζουμε είναι ίσο με 0.01. Ορίζουμε ο αριθμός των εποχών σε κάθε trial να είναι 5.

Preprocessing	Accuracy in val_set
disabled	0.848547844 in epoch = 2
enabled all	0.792735849 in epoch = 2
disabled stopword	0.832419137 in epoch = 2
disabled lemmatize	0.794123989 in epoch = 2
disabled stopword, lemmatize	0.834093666 in epoch = 3

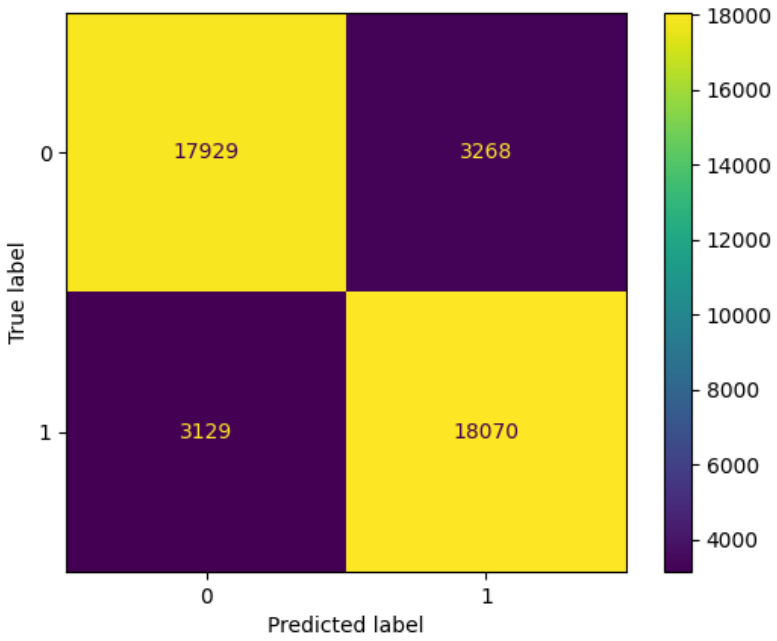
Table 25: Testing preprocessing in final model

Παρατηρούμε πως πράγματι το μοντέλο μας έχει την καλύτερη απόδοση όταν το μοναδικό preprocessing που εφαρμόζουμε είναι εκείνο του tokenizer. Καταλήγουμε, δηλαδή, ότι το καλύτερο μοντέλο μας έχει ως εξής:

Preprocessing	tokenizer's (do_lower_case=True)
Batch size	32
Learning rate	2e-5
Epochs	2
Optimizer	AdamW
Weight decay	0.01
eps	1e-8
Learning Scheduler	yes
L2 Norm Clipping	yes

4.2 Final model evaluation

4.2.1 Confusion matrix



Σχήμα 5: Distilbert model: Confusion matrix

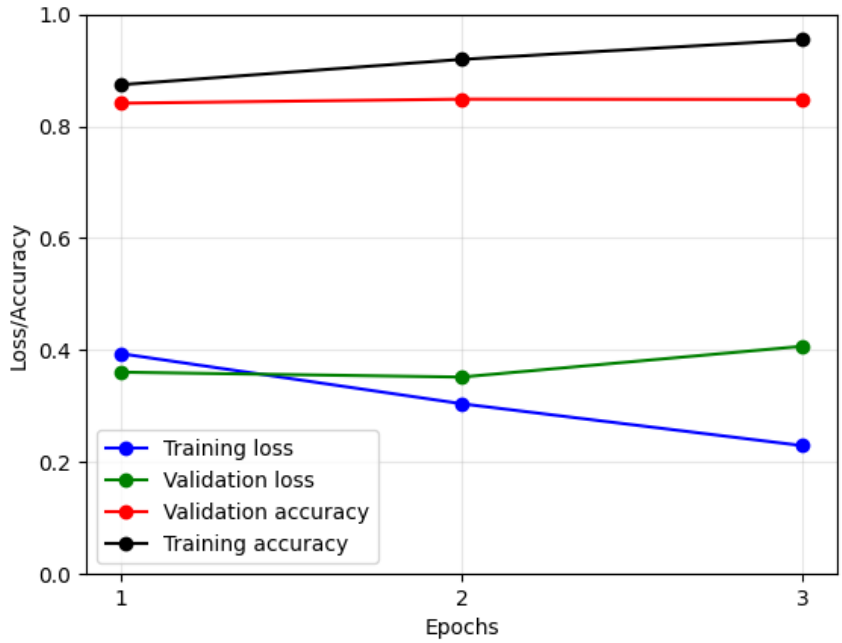
Class	Precision	Recall	F1-score	Support
0	0.85	0.85	0.85	21197
1	0.85	0.85	0.85	21199

Table 26: Distilbert model: Classification report

Το μοντέλο μας πετυχαίνει 85% accuracy. Αποτελεί, δηλαδή, βελτίωση του αρχικού μοντέλου κατά 1%. Όπως και στο BERT, από το classification report βλέπουμε ότι κάθε μετρική έχει την ίδια τιμή και στις δύο κλάσεις. Αυτό δείχνει ότι και αυτό το μοντέλο παρουσιάζει σταθερή απόδοση και στις δύο κλάσεις χωρίς να ευνοεί μια κλάση έναντι της άλλης. Από τα αποτελέσματα στον confusion matrix βλέπουμε ότι το πλήθος των σφαλμάτων του μοντέλου είναι σχεδόν ίδιος (3.268 false positives vs 3.129 false negatives).

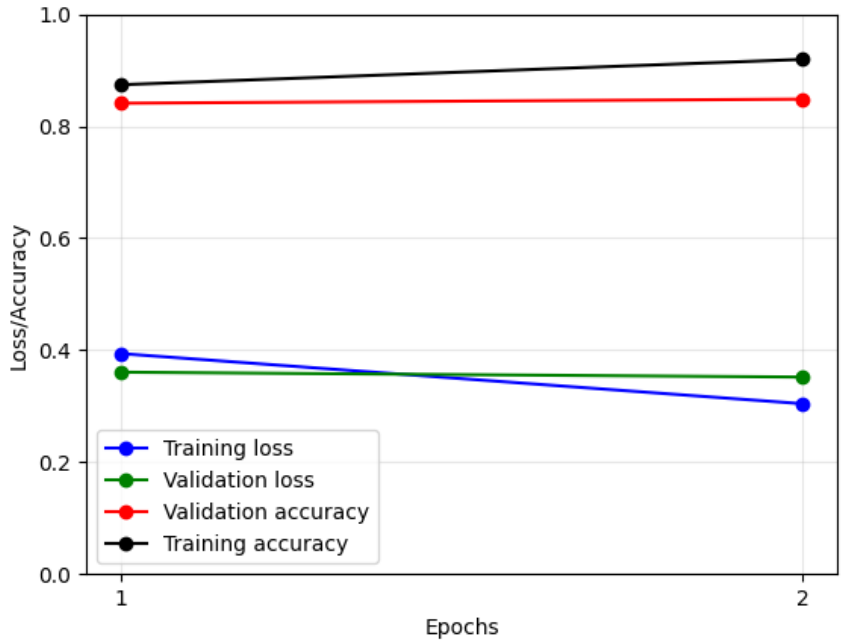
Σε σύγκριση με το BERT, βλέπουμε ότι το BERT ξεπερνά το DistilBert κατά 1% όχι μόνο στο accuracy, αλλά και στις υπόλοιπες μετρικές του classification report, πράγμα που υποδηλώνει για το BERT ελαφρώς καλύτερη ικανότητα ταξινόμησης.

4.2.2 Learning curve



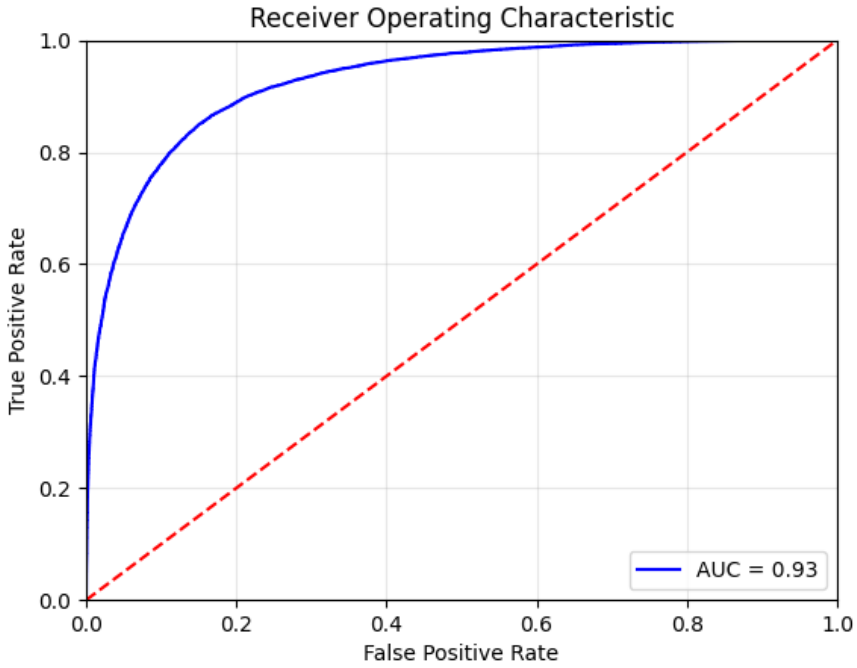
Σχήμα 6: Distilbert model (3 epochs): Learning curve

Μελετώντας τη καμπύλη μάθησης του μοντέλου βλέπουμε ότι η συμπεριφορά του DistlBert είναι πολύ παρόμοια με το BERT. Παρατηρούμε ότι δεν υπάρχει σημαντική βελτίωση του validation accuracy με το πέρασμα των εποχών, ενώ για το training accuracy υπάρχει αύξηση. Ταυτοχρόνως, βλέπουμε ότι από την εποχή 2 και μετά υπάρχει απόκλιση του training loss vs validation loss, γεγονός που υποδεικνύει ότι το μοντέλο μας από την εποχή 2 και μετά κάνει overfit. Μέσω της καμπύλης μάθησης του μοντέλου αιτιολογούμε το early stopping που συμβαίνει στην εποχή 3 (με patience = 2), δίνοντας μας ένα τελικό (βέλτιστο) μοντέλο μετά από μόλις 2 εποχές (Σχήμα 7).
Σημειώνουμε ότι το τελικό μοντέλο του σχήματος 7 πετυχαίνει accuracy στο test set ίσο με 0.84848.



Σχήμα 7: Distilbert **final** model (2 epochs): Learning curve

4.2.3 ROC curve



Σχήμα 8: Distilbert model: ROC curve

Αναφορικά με τη Reciever Operating Characteristic Curve (ROC) του τελικού μοντέλου, παρατηρούμε ότι η τιμή της Area Under the Curve (AUC) ταυτίζεται με τη τιμή

της AUC στο BERT (0.93). Συμπεραίνουμε, δηλαδή, ότι και το εν προκειμένω Distilbert μοντέλο έχει την ικανότητα να διακρίνει με υψηλή αξιοπιστία τις δύο κλάσεις/labels.

5. Results and Overall Analysis

Μέσω των πειραμάτων που τρέξαμε στην παρούσα εργασία καταφέραμε (τόσο στο BERT όσο και στο DistilBert) να ξεπεράσουμε το validation accuracy που πετυχαίναμε σε κάθε άλλη προηγούμενη εργασία. Είδαμε ότι το BERT είναι ελάχιστα πιο ακριβές (1%) σε σύγκριση με το DistilBert, πράγμα που αιτιολογείται δεδομένου ότι το BERT απαιτεί περισσότερους υπολογιστικούς πόρους και χρόνο, χωρίς αυτό να σημαίνει ότι το DistilBert δεν είχε και αυτό εξίσου πολύ καλή απόδοση. Είναι ιδιαίτερα εντυπωσιακό το γεγονός ότι το DistilBERT, παρά τον μειωμένο χρόνο εκπαίδευσης, κατάφερε να πετύχει αποτελέσματα τόσο κοντά σε αυτά του BERT. Επίσης, είναι αρκετά εντυπωσιακό πως μόλις 2 εποχές αρκούσαν για να πετύχει κανείς ένα αρκετά αξιοπρεπές μοντέλο, τόσο στο BERT όσο και στο DistilBert.