ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ ΙΙ

ΕΡΓΑΣΙΑ 4

Πρόλογος

Σκοπός της εργασίας αυτής είναι να προπονηθεί ένα νευρωνικό δίκτυο και συγκεκριμένα ένα νευρωνικό δίκτυο βασισμένο στην αρχιτεκτονική των **transformers**. Το δίκτυο αυτό ονομάζεται **BERT** και χρησιμοποιεί μόνο την δομή του **encoder** από τους transformers για να παράξει **ενισχυμένες αναπαραστάσεις** για κάθε token που δέχεται στην είσοδο του. Αυτές οι αναπαραστάσεις προκύπτουν δηλαδή κυρίως από την διαδοχική εφαρμογή του **attention μηχανισμού** (με πολλαπλά heads) από layer σε layer ώστε επιτευχθεί η βαθιά κατανόηση των συσχετίσεων των λέξεων της εισόδου. Οπότε οι πλούσιες αυτές αναπαραστάσεις που εξάγει αυτό το μοντέλο θα χρησιμοποιηθούν για την επίλυση δύο γνωστών tasks. Το ένα αποτελεί την επίλυση ενός προβλήματος **κατηγοριοποίησης πολλαπλών κλάσεων** με σκοπό την περαιτέρω ανάλυση του χαρακτήρα (**sentiment analysis**) κάποιων αναρτήσεων του twitter και το άλλο αφορά την **απάντηση ερωτήσεων** του dataset **SQuAD 2.0** (και άλλων γνωστών datasets που μετασχηματίζονται σε αυτό) πάνω σε ένα δοσμένο κείμενο. Μιας και η φύση του κάθε task είναι διαφορετική πρέπει να εφαρμοστεί κατάλληλο **fine tuning** δηλαδή επιπλέον επαναλήψεις βασισμένες στο αντίστοιχο task ώστε το μοντέλο να μπορέσει να το μάθει εύκολα μιας και είναι ήδη **pretrained** πάνω σε εκατομμύρια δεδομένα που βρίσκονται στο internet. Η διαδικασία δηλαδή του pretraining έχει ήδη γίνει και χρησιμοποίησε δύο τεχνικές το **Masked Language Model** και το **Next Sentence Prediction** όπου η πρώτη αφορά την πρόβλεψη λέξεων που αφαιρούνταν τυχαία με μια πιθανότητα και η δεύτερη την πρόβλεψη για το αν η επόμενη πρόταση έπεται πράγματι της πρώτης μιας και το μοντέλο δέχεται και ζευγάρια προτάσεων όπως θα φανεί και στο δεύτερο task. Στην συνέχεια, θα αναλυθούν λεπτομερώς τόσο τα βήματα που ακολουθήθηκαν για την διαδικασία του fine tuning, πάνω στην βασική έκδοση του BERT (**BERT-base**) σε κάθε task και παρέχεται από την βιβλιοθήκη του Hugging Face, όσο και κάποιες παρατηρήσεις αλλά και συγκρίσεις μεταξύ διαφορετικών πρακτικών και παραμέτρων.

Ερώτημα 1

Γενική περιγραφή

Τα αρχικά βήματα που ακολουθήθηκαν είναι αρκετά παρόμοια με αυτά των προηγούμενων εργασιών αλλά με κάποιες τροποποιήσεις. Αρχικά, πραγματοποιείται η προεπεξεργασία των δεδομένων όπως και στις προηγούμενες εργασίες με την μόνη διαφορά πως στην περίπτωση αυτή η προεπεξεργασία είναι **πιο απλή** και αφορά κυρίως την **αφαίρεση του θορύβου** που μπορεί να παρουσιάζεται. Δηλαδή αφαιρούνται τα **urls** και κανονικοποιούνται τα **html tags** αλλά και οι συμπτυγμένες λέξεις (**contractions**) στην πλήρη τους μορφή ώστε να μοιάζει πιο πολύ με ανθρώπινο πραγματικό κείμενο. Επίσης αφαιρούνται κάποια **ειδικά σύμβολα** του twitter και τα **tags** (@) με τα ονόματα των χρηστών. Σε αυτό το σημείο αξίζει να σημειωθεί πως **αφαιρώντας τα tags βελτιώθηκε κατά πολύ η απόδοση** σε σύγκριση με όλα τα μοντέλα των προηγούμενων εργασιών στα οποία μειωνόταν αρκετά πράγμα το οποίο σημαίνει πως όντως το μοντέλο αυτό είναι αρκετά πιο προχωρημένο από όλα τα υπόλοιπα μιας και φαίνεται πως δεν μαθαίνει απλά patterns της εισόδου αλλά κατανοεί περισσότερο το νόημα της. Τέλος, σε αυτό συμβάλλει αρκετά και ο «έξυπνος» tokenizer (**WordPiece**) ο οποίος σπάει καταλλήλως τις λέξεις σε tokens. Ο tokenizer αυτός δεν έχει τεράστιο λεξιλόγιο παρόλα αυτά αυτό είναι αρκετά χρήσιμο μιας και αν δεν υπάρχει κάποια λέξη στο λεξιλόγιο τη σπάει πιθανώς σε κάποια **ρίζα** της και έτσι δεν απαιτείται και lemmatization όπως στις προηγούμενες εργασίες.

Η διαδικασία της φόρτωσης των datasets παραμένει ακριβώς η ίδια με τις προηγούμενες ασκήσεις. Η οπτικοποίηση των στατιστικών που εξάγονται από αυτά έχει αλλάξει λίγο και πιο συγκεκριμένα έχει προστεθεί και ένα γράφημα που παρουσιάζει όλα τα **διαφορετικά μήκη** του training set. Παρατηρείται πως τα περισσότερα μήκη των tweets κυμαίνονται από **0 - 150 χαρακτήρες** πράγμα το οποίο δείχνει πως τα μεγέθη **δεν είναι τεράστια** και αυτό αποτελεί σημαντική πληροφορία για την συνέχεια.

Στο σημείο αυτό αρχίζει ουσιαστικά η διαδικασία για το fine tuning. Αρχικά, γίνεται έλεγχος για το αν η **GPU** είναι διαθέσιμη διότι χωρίς την συμβολή της GPU η διαδικασία του fine tuning είναι ατελείωτη. Στην συνέχεια ορίζονται κάποιες συναρτήσεις για την δημιουργία γραφημάτων ακριβώς με τον ίδιο τρόπο όπως στις προηγούμενες εργασίες. Επίσης, ορίζονται όλοι οι scheldulers που χρησιμοποιήθηκαν στις προηγούμενες εργασίες με την προσθήκη ενός ακόμα που είναι ήδη υλοποιημένος από την βιβλιοθήκη του Hugging Face και ουσιαστικά αυτός πραγματοποιεί το **linear warmup** του learning rate δηλαδή **γραμμική αύξηση** του learning rate μέχρι την αρχική τιμή και έπειτα **γραμμική μείωση** μέχρι το 0.

Στην συνέχεια ορίζονται κάποια πολύ χρήσιμα **δομικά στοιχεία** που είναι απαραίτητα και θα τα χρησιμοποιήσει το μοντέλο παρακάτω. Το πρώτο είναι η δημιουργία ενός νέου **custom dataset** το οποίο αρχικά μετασχηματίζει τα κείμενα των tweets σε έγκυρες **κωδικοποιημένες εισόδους** για το μοντέλο και ύστερα αποθηκεύει τις κωδικοποιήσεις αυτές των tweets (έχουν και διαφορετικά μεγέθη). Ουσιαστικά δημιουργούνται κάποιες εγγραφές (**records**) με την μορφή named tuples, τα οποία είναι και αποδοτικά από άποψη χώρου στην μνήμη μιας και αποτελούν απλά tuples και παράλληλα αποθηκεύουν και πληροφορία σχετικά με το τι περιέχει η κάθε θέση. Οπότε δημιουργείται μια τέτοια εγγραφή αφού κωδικοποιηθεί κάποιο tweet μέσω της χρήσης του **tokenizer** που αναφέρθηκε νωρίτερα. Δηλαδή, για κάθε tweet εξάγεται από τον tokenizer ένα σετ από **input ids** τα οποία αφορούν τα id των tokens που προκύπτουν από τις λέξεις του (+ τα ids των ειδικών tokens **CLS** για την αρχή και **SEP** για το τέλος). Το πλήθος των tokens αυτών μπορεί να περιορίζεται και από το **μέγιστο μήκος** που έχει προκαθοριστεί. Επίσης δημιουργούνται και δύο ακόμα συμπληρωματικές κωδικοποιήσεις τα **token type ids** ή αλλιώς τα **segment ids** που δηλώνουν τη σειρά της πρότασης (σε αυτό το task δίνεται μόνο μια πρόταση οπότε υπάρχει μόνο ένα segment με id = 0) και το **attention mask** το οποίο μοιάζει με την τεχνική που εφαρμόζεται σε έναν **decoder** για να μην λαμβάνονται υπόψη οι επόμενες λέξεις αλλά σε αυτή την περίπτωση χρησιμοποιείται για μην λαμβάνεται υπόψη το **padding** που εφαρμόζεται (στο στάδιο αυτό αρκεί οι τιμές να είναι ίσες με 1 μιας και όλα τα tokens δεν αποτελούν padding και είναι ενεργά). Όσον αφορά το padding προστίθεται **δυναμικά σε κάθε κωδικοποίηση** με βάση το μήκος της **μεγαλύτερης ακολουθίας** σε κάθε batch και όχι εξαρχής για την εξοικονόμηση χώρου και για και για την πιο γρήγορη επεξεργασία μιας και μπορεί να αποφευχθεί το επιπλέον padding αν το μήκος της μεγαλύτερης ακολουθίας είναι μικρότερο του μέγιστου προκαθορισμένου μήκους. Κλείνοντας, σε κάποιες πηγές στο internet αναφέρεται πως το περιττό padding επηρεάζει αρνητικά και τα scores.

Έπειτα ακολουθεί η κλάση του μοντέλου που θα δοκιμαστεί. Η κλάση αυτή διαφέρει κυρίως στα εξής σημεία από τις κλάσεις των μοντέλων των προηγουμένων εργασιών:

- **Αρχικοποίηση**(__init__): Το βήμα αυτό διαφέρει τελείως. Ουσιαστικά ορίζεται ο αριθμός των διαφορετικών κλάσεων και το μοντέλο BERT που θα χρησιμοποιηθεί. Εδώ αξίζει να σημειωθεί πως δεν λαμβάνεται το έτοιμο μοντέλο που προσφέρει η βιβλιοθήκη για classification tasks (BertForSequenceClassification) αλλά χρησιμοποιείται το **γενικό μοντέλο (BertModel)**. Ο λόγος που γίνεται αυτό είναι για να υπάρχει μεγαλύτερη ευελιξία με τον **χειρισμό του output** του μοντέλου. Δηλαδή εκτός από τον βασικό χειρισμό της εξαγωγής της αναπαράστασης-embedding του **CLS token** του τελευταίου layer το οποίο χρησιμοποιείται για το classification,

μιας και εμπεριέχει όλο το νόημα της πρότασης (αφού κάνει attend σε όλα τα υπόλοιπα tokens), δοκιμάστηκαν και κάποιες άλλες παραλλαγές που βασίζονται πάλι στο CLS token. Η πρώτη απλά παίρνει το embedding του CLS token του **προτελευταίου layer** και η δεύτερη τα embeddings του CLS των **τεσσάρων τελευταίων τα οποία συνενώνονται**. Παράλληλα δοκιμάστηκε και μια άλλη ξεχωριστή τεχνική που δεν βασίζεται αποκλειστικά στο CLS token και στην ουσία υπολογίζει ένα νέο embedding από τον **μέσο όρο** των embeddings του **προτελευταίου layer**, το οποίο όπως έχει φανεί στην πράξη παράγει μια αρκετά καλή αναπαράσταση για όλη την πρόταση. Τέλος, ορίζεται ένα **dropout layer** και ένας **classifier** για το τελικό projection στον αριθμό των κλάσεων.

- **Forward pass** (forward): Το βήμα αυτό χρησιμοποιεί τα παραπάνω για να παράξει τα τελικά logits.

Πριν τεθεί το μοντέλο σε εφαρμογή έγιναν επίσης και κάποιες αλλαγές στις συμπληρωματικές συναρτήσεις για την εκτέλεση και αξιολόγηση του μοντέλου και συγκεκριμένα στο **grid search**. Αρχικά, προσαρμόστηκαν οι νέες παράμετροι για το **fune tuning** και επίσης ορίστηκε ο **AdamW** ως optimizer (ο οποίος αποτελεί μια παραλλαγή του γνωστού Adam) όπου ο συγκεκριμένος μπορεί να χειριστεί **πιο αποτελεσματικά το weight decay** (normalization) μιας και ο όρος αυτός δεν προστίθεται στην loss function ώστε να επηρεάζει το **gradient** το οποίο με την σειρά του χρησιμοποιείται στους όρους "συσσώρευσης" m και v αλλά **εφαρμόζεται απευθείας** κατά την ενημέρωση. Τέλος ορίζονται και δύο νέοι τύποι για το **learning rate shelduling** βασισμένοι στο linear warmup schelduling που αναφέρθηκε παραπάνω όπου ο πρώτος λέγεται **Linear** όπου **μειώνει γραμμικά** το learning rate μόνο ενώ ό δεύτερος λέγεται **LinearWarmup** όπου **αυξάνει γραμμικά** το learning rate για το 10% (αυθαίρετη επιλογή) των βημάτων και έπειτα το **μειώνει γραμμικά**.

Αξιολόγηση του Fine Tuning του μοντέλου

Η αξιολόγηση του μοντέλου θα γίνει αρχικά κυρίως ως προς τις διαφορετικές υπερπαραμέτρους που προτείνουν και οι συγγράφεις του BERT για fine tuning μαζί με το αντίστοιχο εύρος τιμών. Επίσης, όλα τα μοντέλα εφαρμόζουν **gradient clipping** με μέγιστη τιμή 1 ώστε να αποφευχθούν τυχόν exploding gradients. Επιπλέον τα tokens για τα tweets περιορίζονται στο να **μην ξεπερνάνε την τιμή 128** όπου φάνηκε να συμπεριφέρεται αρκετά παρόμοια με τις μεγαλύτερες τιμές (σε αυτό οφείλεται και το **μικρό μέγεθος** τους που διαπιστώθηκε παραπάνω).

1) Στην πρώτη περίπτωση ελέγχονται οι τιμές ως προς τη **default στρατηγική** (LAST CLS) για τα διαφορετικά **batch sizes** (16, 32) και τα διαφορετικά **learning rates** (5e-5, 3e-5, 2e-5). Οι συνδυασμοί αυτοί εκτελούνται για **3 epochs** (μεγαλύτερος αριθμός οδηγεί σε αρκετό **overfitting**) ώστε να προκύψουν συμπεράσματα και για τον αριθμό των epochs.

Συμπεράσματα για αριθμό των epochs: Παρατηρήθηκε πως αριθμός μεγαλύτερος του 2 οδηγεί σε τεράστιο **overfitting** (πιθανώς διότι το dataset είναι αρκετά μικρό οπότε είναι αρκετά εύκολο για το μοντέλο να το μάθει). Άρα αρκούν 2 epochs μιας και παράγουν σχεδόν πάντοτε τα καλύτερα scores. Έτσι, τα 2 epochs θα αποτελούν τον κοινό άξονα αξιολόγησης στην συνέχεια.

Συμπεράσματα για τιμές των batch sizes: Παρατηρήθηκε πως η τιμή 32 συμπεριφέρεται αρκετά καλύτερα από την τιμή 16 ως προς τα scores. Επιπλέον διαπιστώθηκε πως τα αποτελέσματα που παράγονται είναι **πιο σταθερά** (stable) από εκτέλεση σε εκτέλεση το οποίο είναι αρκετά λογικό λόγω των λιγότερων αλλά και μεγαλύτερων βημάτων που πραγματοποιούνται.

Συμπεράσματα για τιμές των learning rates: Παρατηρήθηκε πως όλες οι τιμές είναι καλές αλλά οι μικρότερες τιμές 3e-5 και 2e-5 παράγουν λίγο καλύτερα αποτελέσματα.

Οπότε ένας καλός συνδυασμός τιμών αποτελεί ο εξής:

i. Batch size: 32
ii. Learning rate: 3e-5
iii. Epochs : 2

2) Σε δεύτερο στάδιο θα αξιολογηθεί η εφαρμογή των **δύο νέων scheldulers** του learning rate ως προς τον καλύτερο αυτόν συνδυασμό. Αυτό που διαπιστώθηκε είναι πως τα αποτελέσματα τους δεν διαφέρουν αρκετά μεταξύ τους αλλά είναι **ελάχιστα χειρότερα** από την περίπτωση χωρίς schelduling. Αυτό ίσως οφείλεται στο γεγονός πως ο **αριθμός των βημάτων** όπου κάνουν την ενημέρωση είναι **μικρός** οπότε δεν έχουν μεγάλη επίδραση στην συμπεριφορά και στα αποτελέσματα.

3) Σε τρίτο και τελευταίο στάδιο θα αξιολογηθεί η εφαρμογή των **υπολοίπων στρατηγικών** για την εξαγωγή της τελικής αναπαράστασης. Όλες οι στρατηγικές τελικά δεν μπορούν να ξεπεράσουν την default στρατηγική ακόμα και εκείνη στην οποία γίνεται η συνένωση (όπου αντί να είναι η καλύτερη όπως έχουν δείξει πειράματα σε άλλα tasks είναι τελικά η χειρότερη). Άρα ισχύει πως η καλύτερη αναπαράσταση **διαφέρει από task σε task**.

Καλύτερο μοντέλο

Το καλύτερο μοντέλο πρόεκυψε τελικά από τον καλύτερο συνδυασμό της περίπτωσης (1) και όπως φαίνεται ξεπερνάει στην συγκεκριμένη εκτέλεση το **78%** ως προς το accuracy και ως προς το f1 score. Αυτό είναι αρκετά εντυπωσιακό μιας και όλα τα υπόλοιπα μοντέλα των προηγουμένων ασκήσεων έφταναν οριακά μέχρι 72-73%. Τέλος, φαίνεται πως το μοντέλο μπορεί να **διακρίνει** αρκετά καλά τις κλάσεις μεταξύ τους παρόλο που δέχεται ένα **unbalanced dataset** και αυτό μπορεί να φανεί αρκετά καλά και από τα confusion matrices και τα roc curves όπου το **roc auc** ξεπερνάει το **90%** σε κάθε κλάση!

Ερώτημα 2

Γενική περιγραφή

Το ερώτημα αυτό διαφέρει σε μερικά σημεία από το προηγούμενο μιας και το task έχει να κάνει με **question answering**. Το dataset για την ανάπτυξη του μοντέλου για question answering είναι το **SQuAD 2.0**. Στο dataset αυτό υπάρχουν ζεύγη ερωτήσεων και κειμένων τα οποία περιλαμβάνουν σίγουρα την απάντηση στην αντίστοιχη ερώτηση (όπως ακριβώς στο SQuAD 1.0) αλλά ταυτόχρονα υπάρχουν και ζεύγη ερωτήσεων και κειμένων όπου η αντίστοιχη ερώτηση **δεν μπορεί να απαντηθεί**. Συγκεκριμένα, για να κάνουν πιο δύσκολο το dataset οι δημιουργοί του παρέχουν ερωτήσεις «παγίδες» όπου πρακτικά δεν μπορούν να απαντηθούν και αναφέρονται σε **οντότητες** του αντίστοιχου κειμένου οπότε είναι πιο εύκολο για ένα μοντέλο να «μπερδευτεί» και να απαντήσει. Άρα το dataset αυτό έχει σκοπό να διακρίνει κατά πόσο κάποιο μοντέλο κατανοεί το νόημα των ερωτήσεων και των κειμένων και απλά δεν απαντά στην τύχη κάποιο κομμάτι κειμένου (μιας και δεν υπάρχουν πάντοτε απαντήσεις).

Σε πρώτο στάδιο φορτώνονται τα train και dev sets που παρέχονται από τους δημιουργούς. Επειδή το format των αρχείων αυτών είναι σε **JSON** θα πρέπει να μετατραπούν σε κάποια από τις γνωστές δομές των datasets της python. Η πιο γνωστή δομή είναι το **DataFrame** της βιβλιοθήκης pandas η οποία χρησιμοποιήθηκε και στο προηγούμενο ερώτημα. Η δομή αυτή παρέχει και η ίδια μεθόδους για την μετατροπή από JSON όμως ο τρόπος που το JSON αυτών των αρχείων είναι δομημένο δεν είναι ο επιθυμητός. Οπότε το **parsing** του κάθε JSON θα πρέπει να γίνει με τέτοιο τρόπο ώστε να λαμβάνεται η απαραίτητη πληροφορία από κάθε τιμή του κάθε πεδίου. Στην αρχή τα πεδία αυτά ήταν άγνωστα οπότε με σταδιακές εκτυπώσεις έγιναν γνωστά μαζί με το περιεχόμενο τους. Η οργάνωση των πεδίων έχει ως εξής: πρώτα γίνεται διαχωρισμός σε

θεματικές ενότητες και στην συνέχεια χωρίζεται η κάθε μια σε αντίστοιχες **παραγράφους** όπου η κάθε παράγραφος έχει το **κείμενο** (context) και ένα σετ από **ερωτοαπαντήσεις**. Διαχωρίζονται λοιπόν το κείμενο, οι ερωτήσεις και οι απαντήσεις και επειδή ενδέχεται να μην υπάρχει πάντοτε απάντηση γίνεται και ο αντίστοιχος έλεγχος στο πεδίο που το αναγράφει ρητά (**is_impossible**). Οπότε μόνο όταν υπάρχει έστω και μια απάντηση επιλέγεται πάντοτε η **πρώτη** και έπειτα υπολογίζεται η θέση του **τελικού χαρακτήρα** της απάντησης μέσα στο αντίστοιχο κείμενο με βάση την θέση του **αρχικού χαρακτήρα** η οποία δίνεται και αυτή ρητά. Βέβαια παρατηρήθηκε πως μπορεί να μην υπάρχει ευθυγράμμιση (**alignment**) του δοσμένου αρχικού χαρακτήρα για μια ή δύο θέσεις (λόγω κάποιων ειδικών χαρακτήρων) οπότε πραγματοποιείται και η κατάλληλη αυτή **διόρθωση**. Τέλος, δημιουργείται ένα DataFrame με όλα τα απαραίτητα πεδία τα οποία θα χρειαστούν στην συνέχεια και για εξοικονόμηση χώρου αν το dataset αφορά το training set περιλαμβάνει μόνο την κύρια απάντηση (από τι φάνηκε δεν υπάρχουν γενικά παραπάνω από μια απαντήσεις στο training set). Αξίζει να σημειωθεί πως από μια γενική εκτύπωση στο παραγόμενο dataset μπορεί να καταλάβει κάποιος ότι είναι αρκετά «καθαρό» οπότε **δεν χρειάζεται κάποιο επιπλέον καθάρισμα**.

Στην συνέχεια αρχίζει ουσιαστικά η διαδικασία που αφορά το fine tuning. Αρχικά γίνεται έλεγχος για το αν η **GPU** είναι διαθέσιμη διότι χωρίς την συμβολή της GPU η διαδικασία του fine tuning, στο αρκετά μεγάλο αυτό dataset, είναι και εδώ ατελείωτη. Στο συγκεκριμένο ερώτημα χρησιμοποιείται κυρίως, για την εξαγωγή των αποτελεσμάτων που θα παρουσιαστούν παρακάτω, η GPU που προσφέρει η πλατφόρμα **Kaggle** μιας και το χρονικό περιθώριο χρήσης είναι αρκετά μεγαλύτερο ενώ παράλληλα φαίνεται προσφέρει αρκετά πιο γρήγορη (2-3 φορές) GPU από εκείνη του **Colab**.

Έπειτα ορίζονται **δύο μετρικές** οι οποίες είναι αυτές που χρησιμοποιούνται και στο paper του SQuAD για την αξιολόγηση της απόδοσης στο dataset αυτό αλλά μπορεί κανείς να τις χρησιμοποιήσει και γενικότερα σε tasks για question answering . Η πρώτη μετρική ονομάζεται **Exact Match** και πρακτικά είναι αυτό που λέει και το όνομα δηλαδή δύο κείμενα είναι ίδια όταν δεν διαφέρουν σε κανέναν χαρακτήρα. Η δεύτερη μετρική ονομάζεται **f1 score** που πρακτικά είναι λίγο πιο χαλαρή από το Exact Match μιας και υπολογίζει το κλασσικό f1 score με βάση το ποσοστό των κοινών λέξεων/tokens σε κάθε ένα από τα δύο κείμενα (δηλαδή εξάγεται precision και recall και εφαρμόζεται harmonic mean) αλλά επειδή υπάρχει περίπτωση να μην ορίζεται, στις περιπτώσεις αυτές το f1 score είναι 0. Πριν από την εφαρμογή των παραπάνω μετρικών βασική προϋπόθεση αποτελεί η **κανονικοποίηση** των κειμένων που συγκρίνονται δηλαδή θα πρέπει να αφαιρεθούν τα σημεία στίξης, τα άρθρα, τα περιττά κενά και να γίνουν πεζοί όλοι οι χαρακτήρες. Η υλοποίηση των μετρικών μπορεί να βρεθεί στο evaluation

script που δίνει το SQuAD αλλά και στη βιβλιοθήκη Hugging Face όπου χρησιμοποιείται σε αντίστοιχα scripts που αφορούν την εκτέλεση μοντέλων που βασίζονται σε transformers όπως το BERT. Ρίχνοντας μια ματιά σε μοντέλα BERT που δημοσιεύονται στο HuggingFace Model Hub διαπιστώθηκε πως μερικά από τα μοντέλα BERT χρησιμοποιούν αυτό το script. Οπότε για να υπάρξει ένας **κοινός άξονας** ώστε να γίνει και η πιο σωστή αξιολόγηση του μοντέλου στην συνέχεια υλοποιήθηκαν οι μετρικές ως μια παραλλαγή λίγο πιο απλή αλλά αρκετά παρόμοια με αυτή που παρέχεται στο *squad_metrics.py* του Hugging Face και στο evaluation script του SQuAD.

Μετά ορίζεται μια παρόμοια συνάρτηση για την εκτύπωση των **learning curves** όπως και οι ίδιοι **scheldulers** με το προηγούμενο ερώτημα.

Στην συνέχεια ορίζονται κάποια **δομικά στοιχεία** παρόμοια με αυτά του προηγούμενου ερωτήματος που είναι απαραίτητα και θα τα χρησιμοποιήσει το μοντέλο παρακάτω. Το πρώτο είναι και εδώ η δημιουργία ενός νέου **custom dataset** το οποίο είναι στενά συνδεδεμένο με ένα αντίστοιχο αρχικό SQuAD DataFrame. Όπως και στο προηγούμενο ερώτημα κάθε αρχική εγγραφή (**record**) του DataFrame αυτού μετασχηματίζεται σε έγκυρες **κωδικοποιημένες εισόδους** για το μοντέλο όπου ύστερα αποθηκεύονται ως **νέες εγγραφές** στο νέο αυτό custom dataset. Πιο συγκεκριμένα, για κάθε μια αρχική εγγραφή γίνεται η χρήση του tokenizer ο οποίος λαμβάνει την ερώτηση και το κείμενο της έπειτα τα συνενώνει με τη σειρά αυτή διαχωρίζοντας τα με το ειδικό token SEP (βάζοντας και ένα SEP token στο τέλος και ένα CLS στην αρχή) και ταυτόχρονα πραγματοποιεί και την αποκοπή (**truncation**) στο **μέγιστο μήκος** που έχει προκαθοριστεί. Η αποκοπή αυτή εφαρμόζεται πάντα στο κείμενο που **έπεται** της ερώτησης. Ο tokenizer αυτός δημιουργεί μια **ειδική δομή** που περιλαμβάνει τις κωδικοποιήσεις που αναφέρθηκαν και στο προηγούμενο ερώτημα (εδώ απλά τα segment ids είναι ίσα με 0 στις θέσεις των tokens της ερώτησης και ίσα με 1 στις θέσεις των tokens του κειμένου) και επίσης εμπεριέχει και τις αντιστοιχίες (**mapping**) του κάθε token στις θέσεις των αντίστοιχων χαρακτήρων που ξεκινάει και τελειώνει αυτό μέσα στο πραγματικό κείμενο (υπάρχουν και έτοιμες μέθοδοι όπως char_to_token και token_to_chars που υπολογίζουν τις αντιστοιχίες αυτές αλλά η παραπάνω δομή δίνει μεγαλύτερη ευελιξία). Αυτό βοηθάει στο να βρεθούν **όλα τα tokens** του κειμένου που **περιλαμβάνουν την απάντηση** μιας και η κάθε απάντηση δίνεται ως ένα ζευγάρι αρχικού και τελικού χαρακτήρα μέσα στο κείμενο. Οπότε σκοπός είναι να βρεθεί αυτό το σύνολο των tokens που περιλαμβάνουν ακριβώς την απάντηση όμως σε περίπτωση που η απάντηση βρίσκεται **εκτός ορίων** του κειμένου έστω και για έναν χαρακτήρα (π.χ λόγω αποκοπής) η εγγραφή που θα δημιουργηθεί θεωρείται πως **δεν έχει απάντηση**. Όπως έγινε αναφορά και στην αρχή υπάρχουν επίσης και αρχικές εγγραφές που δεν έχουν απάντηση εξαρχής οπότε σε αυτές και στις

προηγούμενες ορίζεται το **CLS token ως αρχικό και τελικό token** της απάντησης (αυτό προτείνουν και εφάρμοσαν οι συγγράφεις του BERT). Αντιθέτως με το προηγούμενο ερώτημα που αποθηκευόταν ένα named tuple για κάθε νέα εγγραφή τώρα **αποθηκεύεται η δομή** του που δίνει ο tokenizer η οποία **έχει εμπλουτιστεί** με τις θέσεις του **αρχικού και του τελικού token** της απάντησης και με το id της αρχικής εγγραφής στο custom dataset το οποίο ταυτίζεται με αυτό στο DataFrame (το named tuple εξακολουθεί να δημιουργείται αλλά δυναμικά όταν πρόκειται να δοθεί η εγγραφή προς τα έξω). Αυτό είναι αρκετά χρήσιμο διότι ορίζεται και μια μέθοδος στο νέο αυτό custom dataset που **δημιουργεί μια έγκυρη απάντηση** για κάθε εγγραφή του, δεδομένης της πληροφορίας που κρατάει το custom dataset για κάθε μια από αυτές. Αναζητά δηλαδή την αντίστοιχη αρχική εγγραφή στο DataFrame για το πραγματικό κείμενο μιας και **στο custom dataset περιλαμβάνονται μόνο κωδικοποιήσεις**. Αυτή η μέθοδος χρησιμοποιείται κυρίως με τις **προβλεπόμενες θέσεις** των tokens που παράγει το μοντέλο οπότε μπορεί να «χτίσει» την απάντηση ακολουθώντας την **ανάποδη διαδικασία** (αφού διατηρούνται τα παραπάνω mappings και οι συσχετίσεις με τις αρχικές εγγραφές). Κλείνοντας, ο τρόπος με τον οποίο εφαρμόζεται το padding είναι παρόμοιος με τον τρόπο του προηγούμενου ερωτήματος.

Έπειτα ακολουθεί η κλάση του μοντέλου που θα δοκιμαστεί. Η κλάση αυτή διαφέρει κυρίως στα εξής σημεία από τη κλάση του προηγούμενου ερωτήματος:

- **Αρχικοποίηση**(__init__): Στο βήμα αυτό αρχικοποιείται το έτοιμο μοντέλο που παρέχει η βιβλιοθήκη Hugging Face για question answering. Ένα **πιθανώς ελάττωμα** που παρατηρήθηκε είναι πως εφαρμόζεται σε όλο το sequence ο start και ο end classifier για την πρόβλεψη του αρχικού και του τελικού token της απάντησης και όχι στο κείμενο μόνο (γι αυτό χρειάζεται κάποιος έλεγχος κατά την δημιουργία της απάντησης ώστε να είναι έγκυρη). Βέβαια αυτό μπορεί να βγάζει κάποιο νόημα μιας και το μοντέλο μπορεί να μάθει επιπλέον πως στη περιοχή της ερώτησης δεν μπορεί να υπάρξει απάντηση.
- **Forward pass** (forward): Το βήμα αυτό χρησιμοποιεί το παραπάνω μοντέλο για να παράξει τη πιθανή αρχή και τέλος της απάντησης αλλά και το μέσο loss.
- **Evaluation ανά batch** (batchEval): Η μέθοδος αυτή **υπολογίζει** και τα scores που προκύπτουν από τις παραπάνω **μετρικές**. Δηλαδή λαμβάνει όλες τις αρχικές απαντήσεις και την απάντηση που έδωσε το μοντέλο για ένα δείγμα στο batch, κάνει ένα **max aggregation** για το δείγμα αυτό και υπολογίζει **στο τέλος** για όλα τα batches και για κάθε μετρική το **μέσο score**. Προκειμένου να βελτιωθούν τα scores, αντί να επιλέγονται κάθε φορά με **αφελή τρόπο** τα tokens της πιο πιθανής αρχής και του πιο πιθανού τέλους που παράγει το

μοντέλο, τα οποία μπορεί να μην δώσουν τελικά μια έγκυρη απάντηση, διερευνούνται οι **k** (νέα υπερπαράμετρος και ορίζεται ίση με 20 σε όλες τις περιπτώσεις) **πιο πιθανές αρχές και τέλη** ώστε να βρεθεί με μεγαλύτερη πιθανότητα μια έγκυρη απάντηση.

- **Προβλέψεις (predict):** Η μέθοδος αυτή παράγει με τον αφελή τρόπο την πιο πιθανή αρχή και τέλος (δεν χρησιμοποιείται)

Ό,τι ακολουθεί μέχρι το τέλος είναι παρόμοιο με ό,τι ακολουθήθηκε στο προηγούμενο ερώτημα.

Αξιολόγηση του Fine Tuning του μοντέλου

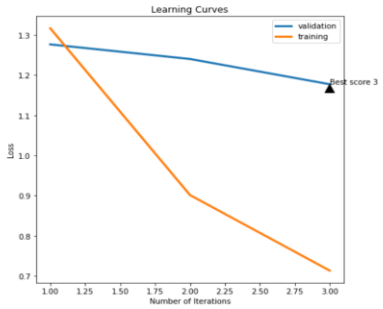
Το fine tuning στο SQuAD 2.0 ήταν μια αρκετά **χρονοβόρα** διαδικασία μιας και το μέγεθος του ήταν αρκετά μεγάλο. Τρέχοντας το με GPU στο Colab ήταν αρκετά αργό οπότε έγινε χρήση της GPU που δίνει το Kaggle και έτσι τα περισσότερα αποτελέσματα παράχθηκαν εκεί. Επειδή όμως υπήρχε όριο με τα sessions δεν υπήρχε κάποιος τρόπος να τρέξουν όλα τα πειράματα σε μια εκτέλεση οπότε θα παρουσιαστούν παρακάτω συγκεκριμένα **στιγμιότυπα** από διαφορετικά πειράματα.

Ένα ακόμα σημαντικό πρόβλημα που παρουσιάστηκε πέρα από τον χρόνο εκτέλεσης ήταν και ο περιορισμένος χώρος μνήμης. Όσο πιο πολύ μεγάλωνε το μέγεθος των ζευγαριών ερώτησης και κειμένου που δίνονταν στο μοντέλο και όσο μεγάλωνε και το batch size το session σταμάταγε λόγω **υπερβολικής χρήσης της RAM** (κυρίως στο Colab) οπότε παρουσιαζόταν ένα trade-off ανάμεσα στις δύο αυτές παραμέτρους. Οι καλύτερες τιμές που δεν δημιουργούσαν πρόβλημα ήταν να **περιοριστεί το μέγεθος στη τιμή 256 όταν το batch size ήταν 16 και στη τιμή 384 όταν το batch size ήταν 8**. Μεγαλύτερα batch sizes όπως 32 ήταν αδύνατο να τρέξουν. Από την θεωρία μεγάλα datasets είναι **καλό να τρέχουν με μεγάλα batch sizes και μικρά learning rates** (για να μην γίνονται τεράστια βήματα) αλλά δυστυχώς δεν δινόταν η δυνατότητα δοκιμής μιας τέτοιας περίπτωσης.
Σε πρώτη φάση έγινε δοκιμή του μοντέλου με :

i. Batch size: 8
ii. Learning rate: 5e-5
iii. Epochs : 3
iv. Χωρίς schelduler

Όπως φαίνεται από το παρακάτω στιγμιότυπο παρατηρήθηκε μια **αρκετά αργή μείωση του validation loss** ξεκινώντας **από μια αρκετά υψηλή τιμή** και παράλληλα το **overfitting** ήταν τεράστιο. Αυτό είχε ως αντίχτυπο και στα αποτελέσματα των μετρικών απόδοσης όπου το Exact Match φτάνει οριακά στο 70% και το f1 score στο 72% μετά από 3 epochs.

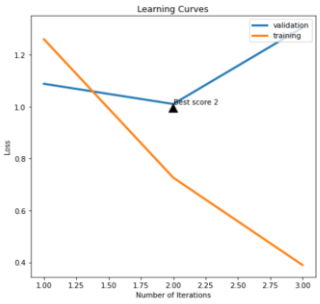
```
Grid Search 1 Batch size: 8, Learning rate: 5e-05, Schelduler: None

Epoch: 1

Batch Training: 100%|██████████| 16290/16290 [1:06:33<00:00,  4.08it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:34<00:00,  3.76it/s]

Training: Loss = 1.31671
Validation: Loss = 1.27674, Exact match = 0.63682, F1 score = 0.67294

Epoch: 2

Batch Training: 100%|██████████| 16290/16290 [1:06:37<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:34<00:00,  3.77it/s]

Training: Loss = 0.90118
Validation: Loss = 1.24039, Exact match = 0.67456, F1 score = 0.70723

Epoch: 3

Batch Training: 100%|██████████| 16290/16290 [1:06:38<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:33<00:00,  3.77it/s]

Training: Loss = 0.71298
Validation: Loss = 1.17746, Exact match = 0.69536, F1 score = 0.72407
```



Η πρώτη σκέψη ήταν πως μάλλον ο **schelduler πρέπει να ήταν απαραίτητος** αφού μετά από μια ματιά στο *run_squad.py* του Hugging Face χρησιμοποιείται **linear decay του learning rate ως default τιμή.** Το BERT εξάλλου χρησιμοποιούσε και αυτό linear decay του learning rate στο μεγαλύτερο αριθμό των βημάτων του κατά το pretraining. Από την θεωρία επίσης είναι γνωστό πως οι learning scheldulers αποτελούν μια πολύ καλή τεχνική για **δυναμική προσαρμογή του learning rate**. Έχοντας ένα **σταθερό learning rate** επίσης για ένα μεγάλο πλήθος βημάτων υπάρχει περίπτωση να **απομακρύνεται το loss από το ελάχιστο**. Όπως φαίνεται στα δύο πρώτα στιγμιότυπα παρακάτω με την εφαρμογή του linear decay shelduler τα αποτελέσματα φαίνεται να είναι με διαφορά αρκετά καλύτερα από το πρώτο κιόλας epoch. Φαίνεται για την **μεγαλύτερη** (5e-5) και την **μεσαία τιμή** (3e-5) learning rate πως το **validation loss είναι αρκετά μικρό** ειδικά στο δεύτερο epoch με το Exact Match score να ξεπερνάει το 73% και με το f1 score να ξεπερνάει το 76% πράγμα το οποίο είναι αρκετά **κοντά στα αποτελέσματα του paper** και ξεπερνάει και τα περισσότερα **αποτελέσματα** παρόμοιων μοντέλων που είναι **δημοσιευμένα στο Hugging Face**. Βέβαια η τελευταία περίπτωση με το **μικρότερο** (2e-5) learning rate φαίνεται να μην τα πηγαίνει και τόσο καλά με τα learning curves αλλά στο τρίτο epoch και αυτή πιάνει καλά scores στις μετρικές. Όλες οι περιπτώσεις όμως **φαίνεται να μην χρειάζονται 3 epochs** μιας και το validation loss εκτοξεύεται. Οπότε η παραπάνω περίπτωση **με 2 epochs** και **linear schelduler** φαίνεται μέχρι στιγμής να αποτελεί αρκετά **καλή επιλογή** με μικρό βαθμό overfitting.

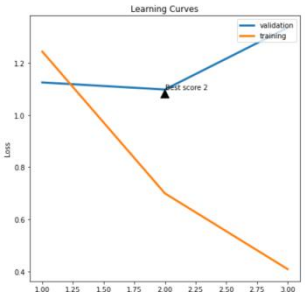
Grid Search 1 Batch size: 8, Learning rate: 5e-05, Schelduler: Linear

```
Epoch: 1

Batch Training: 100%|██████████| 16290/16290 [1:06:42<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:22<00:00,  3.89it/s]

Training: Loss = 1.25916
Validation: Loss = 1.08763, Exact match = 0.70134, F1 score = 0.73300

Epoch: 2

Batch Training: 100%|██████████| 16290/16290 [1:06:41<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:21<00:00,  3.89it/s]

Training: Loss = 0.72614
Validation: Loss = 1.00980, Exact match = 0.73587, F1 score = 0.76534

Epoch: 3

Batch Training: 100%|██████████| 16290/16290 [1:06:44<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:24<00:00,  3.86it/s]

Training: Loss = 0.38959
Validation: Loss = 1.30432, Exact match = 0.72728, F1 score = 0.76083
```



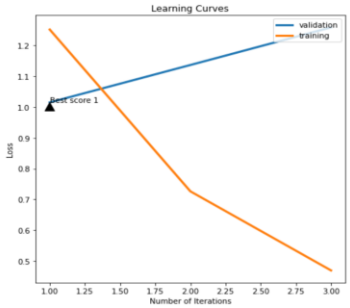
Grid Search 2 Batch size: 8, Learning rate: 3e-05, Schelduler: Linear

```
Epoch: 1

Batch Training: 100%|██████████| 16290/16290 [1:06:44<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:27<00:00,  3.83it/s]

Training: Loss = 1.24275
Validation: Loss = 1.12444, Exact match = 0.70403, F1 score = 0.73195

Epoch: 2

Batch Training: 100%|██████████| 16290/16290 [1:06:43<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:23<00:00,  3.87it/s]

Training: Loss = 0.69981
Validation: Loss = 1.09745, Exact match = 0.72896, F1 score = 0.75880

Epoch: 3

Batch Training: 100%|██████████| 16290/16290 [1:06:39<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:24<00:00,  3.86it/s]

Training: Loss = 0.40988
Validation: Loss = 1.33412, Exact match = 0.72619, F1 score = 0.75902
```



Grid Search 1 Batch size: 8, Learning rate: 2e-05, Schelduler: Linear

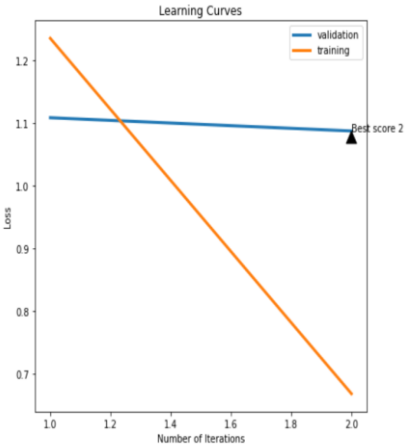
```
Epoch: 1

Batch Training: 100%|██████████| 16290/16290 [1:06:34<00:00,  4.08it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:38<00:00,  3.73it/s]

Training: Loss = 1.25178
Validation: Loss = 1.01509, Exact match = 0.71886, F1 score = 0.74828

Epoch: 2

Batch Training: 100%|██████████| 16290/16290 [1:06:38<00:00,  4.07it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:40<00:00,  3.71it/s]

Training: Loss = 0.72611
Validation: Loss = 1.13666, Exact match = 0.72534, F1 score = 0.75750

Epoch: 3

Batch Training: 100%|██████████| 16290/16290 [1:06:34<00:00,  4.08it/s]
Batch Evaluation: 100%|██████████| 1485/1485 [06:42<00:00,  3.69it/s]

Training: Loss = 0.46914
Validation: Loss = 1.25880, Exact match = 0.73099, F1 score = 0.76377
```



Στην συνέχεια εκτελέστηκαν τα ίδια πειράματα με **batch size 16** για 2 epochs (σύμφωνα με την παραπάνω παρατήρηση).

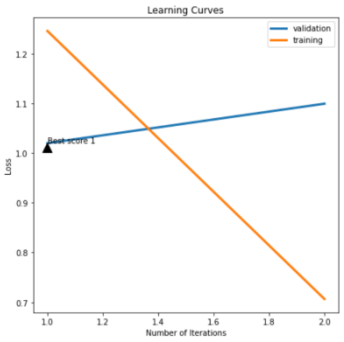
Grid Search 1 Batch size: 16, Learning rate: 5e-05, Schelduler: Linear

```
Epoch: 1
Batch Training: 100%|████████| 8145/8145 [2:56:46<00:00,  1.30s/it]
Batch Evaluation: 100%|████████| 743/743 [11:41<00:00,  1.06it/s]
Training: Loss = 1.23484
Validation: Loss = 1.10826, Exact match = 0.69822, F1 score = 0.72654

Epoch: 2
Batch Training: 100%|████████| 8145/8145 [2:58:02<00:00,  1.31s/it]
Batch Evaluation: 100%|████████| 743/743 [11:32<00:00,  1.07it/s]
Training: Loss = 0.66844
Validation: Loss = 1.08711, Exact match = 0.71743, F1 score = 0.74828
```



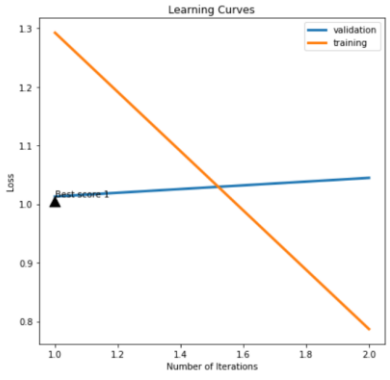
Grid Search 1 Batch size: 16, Learning rate: 3e-05, Schelduler: Linear

```
Epoch: 1

Batch Training: 100%|████████| 8145/8145 [54:24<00:00,  2.50it/s]
Batch Evaluation: 100%|████████| 743/743 [06:34<00:00,  1.88it/s]

Training: Loss = 1.24525
Validation: Loss = 1.01972, Exact match = 0.71119, F1 score = 0.73938

Epoch: 2

Batch Training: 100%|████████| 8145/8145 [54:22<00:00,  2.50it/s]
Batch Evaluation: 100%|████████| 743/743 [06:36<00:00,  1.87it/s]

Training: Loss = 0.70640
Validation: Loss = 1.09908, Exact match = 0.71650, F1 score = 0.74858
```



Grid Search 2 Batch size: 16, Learning rate: 2e-05, Schelduler: Linear

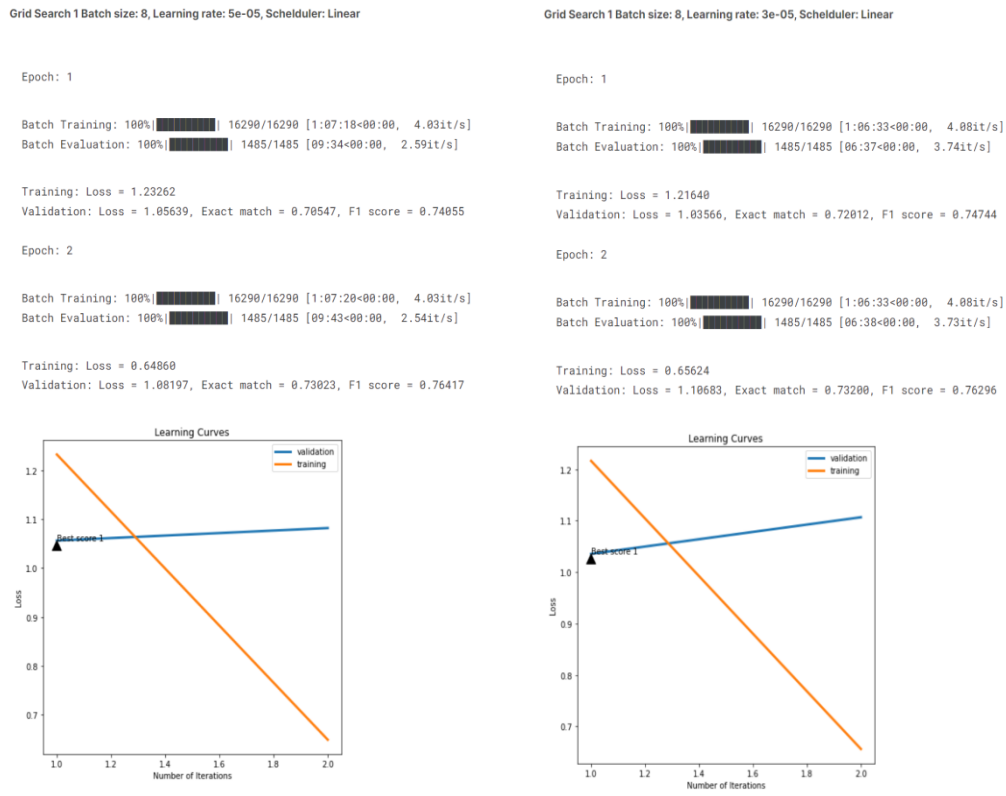
```
Epoch: 1

Batch Training: 100%|████████| 8145/8145 [54:16<00:00,  2.50it/s]
Batch Evaluation: 100%|████████| 743/743 [06:35<00:00,  1.88it/s]

Training: Loss = 1.29211
Validation: Loss = 1.01290, Exact match = 0.70985, F1 score = 0.74042

Epoch: 2

Batch Training: 100%|████████| 8145/8145 [54:19<00:00,  2.50it/s]
Batch Evaluation: 100%|████████| 743/743 [06:38<00:00,  1.86it/s]

Training: Loss = 0.78687
Validation: Loss = 1.04464, Exact match = 0.71844, F1 score = 0.74994
```



Από τις παραπάνω περιπτώσεις φαίνεται πως το **μεγαλύτερο batch size παράγει χειρότερα αποτελέσματα** . Σε αυτό πιθανώς να οφείλεται το γεγονός πως **έχει μικρύνει αρκετά το μέγιστο μήκος** του κάθε ζευγαριού ερώτησης και κειμένου που σημαίνει πως σημαντική πληροφορία έχει χαθεί. Παρόλα αυτά τα αποτελέσματα και εδώ είναι **αρκετά ικανοποιητικά** και επιβεβαιώνεται πως μεγαλώνοντας το batch size τα μικρότερα learning rates είναι και καλύτερα ως προς το validation loss. (Επίσης μπορεί να διαπιστώσει κανείς τη σημαντική διαφορά στον χρόνο εκτελώντας με GPU στο Colab βλέποντας τη πρώτη περίπτωση)

Καλύτερο μοντέλο

Το καλύτερο μοντέλο προέκυψε από την επιλογή ενός εκ των δύο καλύτερων μοντέλων που παρουσιάζονται στην συνέχεια. Και οι δύο επιλογές είναι αρκετά καλές στα scores (πάνω από 73% Exact Match και πάνω από 76% f1) αλλά η καλύτερη ήταν η πρώτη μιας και το validation loss διατηρείται πιο χαμηλό:



Όπως φαίνεται το **validation loss αυξήθηκε ελάχιστα** και αυτό οφείλεται μάλλον στο γεγονός πως με την αφαίρεση ενός epoch η μείωση του learning rate από τον schelduler **γίνεται πιο απότομα**.

Ερώτημα 3

Γενική περιγραφή

Το ερώτημα αυτό αποτελεί ουσιαστικά συνέχεια του προηγούμενου ερωτήματος. Ειδικότερα, θα γίνει η εφαρμογή της αρχιτεκτονικής αυτής που αναπτύχθηκε και παρουσιάστηκε στο προηγούμενο ερώτημα **σε επιπλέον datasets** που σχετίζονται και αυτά με question answering, όπως και το SQuAD 2.0. Τα datasets αυτά είναι το TriviaQA, το NQ, το QuAC, και το NewsQA και ο κύριος λόγος που επιλέχθηκαν είναι διότι έχουν αρκετές ομοιότητες με το SQuAD 2.0 αφού αντιμετωπίζουν το πρόβλημα του question answering ως ένα πρόβλημα κατανόησης κειμένου (**reading comprehension**). Δηλαδή και σε αυτά τα datasets θα πρέπει να βρεθεί η απάντηση μέσα στο κείμενο με βάση κάποια σχετική ερώτηση. Ο λόγος εισαγωγής των επιπλέον αυτών datasets είναι η περαιτέρω ανάλυση της **αποτελεσματικότητας των μοντέλων** πάνω σε question answering και συγκεκριμένα το μοντέλο αυτό που θα αναλυθεί είναι το μοντέλο BERT που αναπτύχθηκε προηγουμένως. Εμβαθύνοντας, θα εξεταστεί η ικανότητα **γενίκευσης** του μοντέλου **πέρα από τα όρια του dataset που έχει γίνει fine tuned** και γι αυτό θα εξεταστεί και **πάνω στα υπόλοιπα datasets** μιας και η φύση τους είναι παρόμοια αλλά το περιεχόμενο και η δομή των ερωτοαπαντήσεων παρουσιάζουν κάποιες σημαντικές διαφορές και ποικιλομορφία. Οπότε έτσι θα γίνει και αντιληπτό για το αν το μοντέλο όντως κατανοεί τις ερωτήσεις και το κείμενο για να παράξει την απάντηση και δεν παράγει τις απαντήσεις με βάση κάποια heuristics και οντότητες που μαθαίνει από το dataset που γίνεται fine tuned. Όλα τα παραπάνω περιγράφονται αναλυτικά στο paper *What do Models Learn from Question Answering Datasets?*. Σκοπός του ερωτήματος αυτού είναι η δημιουργία όλων των παραπάνω datasets και η προσέγγιση του πίνακα 3 του αντίστοιχου paper με βάση το μοντέλο του προηγούμενου ερωτήματος και ενός μοντέλου που προσεγγίζει αυτό που χρησιμοποιήθηκε στο paper.

Η κύρια επέκταση στο προηγούμενο ερώτημα είναι η διαδικασία δημιουργίας των datasets και η διαδικασία αξιολόγησης. Για να μπορέσουν να εξεταστούν με έναν κοινό τρόπο **μετατρέπονται όλα** στην ίδια ακριβώς μορφή που έχει το **SQuAD 2.0**. Αυτό επιτυγχάνεται με βάση τους μετατροπείς (**converters**) που προσφέρονται από τους συγγραφείς του παραπάνω paper σε ένα GitHub repository, όπου όλα τα βήματα περιγράφονται αναλυτικά εκεί. Οπότε με βάση αυτά τα βήματα, τα οποία διαφέρουν από dataset σε dataset, δημιουργήθηκαν και τα αντίστοιχα **bash scripts** όπου τα εκτελούν με αυτοματοποιημένο τρόπο. Το κάθε script αναφέρεται σε ένα dataset και έχει ενσωματωθεί **μέσα στο ίδιο το notebook** (λαμβάνοντας και κάποιες εξωτερικές παραμέτρους από το περιβάλλον). Παρόλο που η διαδικασία μετατροπής μπορεί να γίνει και μέσα από το notebook επειδή σε κάποια datasets είναι **αρκετά χρονοβόρα** (π.χ στο TriviaQA ο χρόνος μετατροπής ήταν ~ 15 λεπτά

ενώ στο NQ > 2 ώρες) τα datasets αυτά μετατράπηκαν μια φορά και στην συνέχεια ανέβηκαν στο **Google Drive**. Οπότε προστέθηκαν και **επιπλέον bash scripts** σε κάθε dataset ώστε να **φορτώθουν τα αρχεία** αυτά που ανέβηκαν στο Google Drive. Επίσης στο Google Drive ανέβηκε και απαραίτητο υλικό για την μετατροπή του NewsQA dataset όπου απαιτούσε μια ειδική διαχείριση (χρήση docker) για την παραγωγή του η οποία έγινε εκτός του notebook.

Όταν λοιπόν τα datasets έχουν μετατραπεί δηλαδή όταν έχουν μετατραπεί τα αντίστοιχα training και dev sets τους δημιουργείται ένα **DataFrame** για το καθένα και προβάλλεται ένα μέρος τους. Η διαδικασία δημιουργίας του DataFrame ήταν ακριβώς η ίδια με προηγουμένως με την μόνη διαφορά πως υπάρχουν απαντήσεις που η αρχική τους θέση να **μην ευθυγραμμίζεται** μέσα στο κείμενο **έως και 3 χαρακτήρες πίσω ή μπροστά** οπότε και εδώ γίνεται διόρθωση. Επίσης εμφανίζονται και τα μεγέθη που έχει το καθένα DataFrame (που αφορά κάποιο training ή dev set) και όπως φαίνεται τα μεγέθη για τα περισσότερα datasets είναι **ίδια ή διαφέρουν για ελάχιστες παρατηρήσεις** σε σύγκριση με τα μεγέθη που αναφέρονται στο paper. Εκεί που η διαφορά στον αριθμό των παρατηρήσεων γίνεται αισθητή είναι στο NewsQA dataset. Παρόλα αυτά τα μεγέθη του training και του dev set του NewsQA που θα χρησιμοποιηθούν στην συνέχεια (92549 και 5166) φαίνεται να είναι έγκυρα μιας και τα ίδια μεγέθη αναφέρονται και στο NewsQA dataset που παρέχει το Hugging Face. Πιθανώς οι συγγραφείς να ακολούθησαν κάποια διαφορετική προσέγγιση ή το dataset όταν εκτελούσαν τα πειράματα να είχε διαφορετικό μέγεθος.

Στην συνέχεια θα αναλυθούν κάποιες τροποποιήσεις ή μικρές προσθήκες που έγιναν για να επιλυθούν κάποια ζητήματα.

Η πρώτη αλλαγή είχε να κάνει με την αντιμετώπιση προβλημάτων που σχετίζονται με τον χώρο μνήμης. Ενώ στο προηγούμενο ερώτημα δεν παρουσιαζόταν κάποιο πρόβλημα, σε αυτή την περίπτωση που διατηρούνται **ταυτόχρονα παραπάνω από ένα datasets** στη μνήμη φάνηκε πως πρέπει να γίνει μια αποτελεσματικότερη διαχείριση σχετικά με το τι θα πρέπει να αποθηκεύεται. **Η δομή των encodings** του tokenizer κατανάλωνε τελικά **αρκετό χώρο** μιας και αποθηκευόταν περιττή πληροφορία που δεν χρειαζόταν κυρίως στις εγγραφές του training set. Οπότε ο πιο απλός τρόπος να αντικατασταθεί η δομή αυτή ήταν να αποθηκεύεται η σημαντική πληροφορία απευθείας (σε named tuples) και η επιπλέον πληροφορία να αποθηκεύεται ξεχωριστά όταν αυτή είναι απαραίτητη για την συνέχεια. Γι αυτόν τον λόγο **οι αντιστοιχίσεις των tokens** στους χαρακτήρες του κειμένου αποθηκεύονται **μόνο για τα dev sets** όπου εκεί είναι κυρίως απαραίτητες αφού πάνω στα dev sets δημιουργούνται οι απαντήσεις που προβλέπει το μοντέλο για την εξαγωγή των αποτελεσμάτων των μετρικών.

Η δεύτερη αλλαγή είχε να κάνει με τον τρόπο που θα παρουσιάζονταν τα αποτελέσματα. Το **fine tuning** πάνω σε κάθε dataset ήταν μια **αρκετά χρονοβόρα διαδικασία** οπότε ήταν αδύνατο να παρουσιαστεί ένα ενιαίο αποτέλεσμα. Η πρώτη ιδέα ήταν να γίνει αρχικά αποθήκευση των παραμέτρων του καθενός fine tuned μοντέλου και ύστερα να ανέβουν παρομοίως στο Google Drive. Όμως αφενός το μέγεθος των παραμέτρων θα ήταν αρκετά μεγάλο αφετέρου θα έπρεπε να προστεθούν επιπλέον scripts. Έπειτα από αρκετή έρευνα για εναλλακτικές λύσεις βρέθηκε μια αρκετά απλή η οποία δίνει το ίδιο το Hugging Face. Πιο συγκεκριμένα, το Hugging Face δίνει την **δυνατότητα ανεβάσματος των μοντέλων σε προσωπικά repositories**. Έτσι, δημιουργήθηκε ένας προσωπικός λογαριασμός στο Hugging Face ώστε να ανεβαίνουν εκεί τα διαφορετικά fine tuned μοντέλα. Οπότε στο τέλος του fine tuning προστέθηκε η **δυνατότητα αποθήκευσης** του μοντέλου τοπικά ώστε να γίνει έπειτα το **ανέβασμα στο προσωπικό repository**. Τέλος, το τεράστιο πλεονέκτημα ήταν πως **εφαρμόζεται πολύ απλά** στην τρέχουσα υλοποίηση μιας και αντί να δίνεται ως παράμετρος να φορτώνεται το pretrained bert base uncased τώρα δίνεται το όνομα του repository ώστε να φορτώνεται από εκεί το fine tuned μοντέλο.

Κλείνοντας, προστέθηκαν και κάποιες βοηθητικές συναρτήσεις για το **fine tuning** των μοντέλων των διαφορετικών datasets και για την **αξιολόγηση** τους.

Αξιολόγηση του Fine Tuning των μοντέλων

Η αξιολόγηση των μοντέλων των διαφορετικών datasets θα γίνει με βάση την διαδικασία που ακολούθησαν και οι συγγραφείς του paper. Πιο συγκεκριμένα, θα παρουσιαστούν τα αποτελέσματα των μοντέλων **μόνο** ως προς το **f1 score** σε έναν πίνακα παρόμοιο με τον πίνακα 3 του paper ώστε να μπορεί να γίνει και η σύγκρισή τους. Δηλαδή θα παρουσιαστούν τα f1 scores των μοντέλων που έχουν γίνει fine tuned σε ένα συγκεκριμένο dataset έναντι όλων των datasets.

Σε πρώτο στάδιο θα πρέπει να γίνει η επιλογή των **κατάλληλων υπερπαραμέτρων** του κάθε μοντέλου που θα αναπτυχθεί πάνω στο κάθε ένα dataset. Στο προηγούμενο ερώτημα βρέθηκε ένας αρκετά καλός συνδυασμός υπερπαραμέτρων για το SQuAD 2.0 οπότε θα εφαρμοστεί αυτός ο συνδυασμός και στα μοντέλα των υπόλοιπων datasets ώστε να δοκιμαστεί η **αποτελεσματικότητα** του και σε αυτά. Παρόλα αυτά οι συγγραφείς ακολουθούν έναν διαφορετικό συνδυασμό υπερπαραμέτρων ο οποίος βασίζεται σε ένα **αρκετά μεγαλύτερο batch size** (24) και **μικρότερο learning rate** (3e-5). Οι συγγραφείς επιπλέον χρησιμοποιούν 16GB RAM για τρέξουν τα πειράματα τους κάτι το οποίο δεν παρέχει το Colab. Όμως διαπιστώθηκε πως το Kaggle δίνει τελικά αυτή την δυνατότητα. Οπότε για να υπάρξει ένας κοινός άξονας θα παρουσιαστούν αποτελέσματα ακολουθώντας μια προσέγγιση παρόμοια με αυτή που ακολούθησαν και οι συγγραφείς. Έτσι, όλα τα πειράματα που θα παρουσιαστούν στην συνέχεια διεξάχθηκαν στο Kaggle.

1) Καλύτερος συνδυασμός προηγούμενου ερωτήματος:

i. Batch size: 8
ii. Learning rate: 5e-5
iii. Epochs : 2
iv. Max seq length: 384
v. Doc stride: 0
vi. Schelduler: Linear

	SQuAD	TriviaQA	NQ	QuAC	NewsQA
SQuAD	0.768707	0.453962	0.471520	0.208758	0.349170
TriviaQA	0.406863	0.616261	0.469505	0.198951	0.222970
NQ	0.511162	0.449887	0.732137	0.215928	0.175460
QuAC	0.309943	0.354544	0.336785	0.349126	0.210997
NewsQA	0.376033	0.493506	0.477060	0.174142	0.533121

2) Συνδυασμός παρόμοιος με αυτόν που ακολούθησαν οι συγγραφείς:

i. Batch size: 24
ii. Learning rate: 3e-5
iii. Epochs : 2
iv. Max seq length: 384
v. Doc stride: 0 (οι συγγραφείς χρησιμοποιούν την τιμή 128 όμως δεν ήταν ξεκάθαρο πως ακριβώς χρησιμοποιήθηκε αυτή η υπερπαράμετρος)
vi. Schelduler: Linear (δεν αναγράφεται αν έγινε χρήση κάποιου schelduler όμως πιθανώς να χρησιμοποιήθηκε)

	SQuAD	TriviaQA	NQ	QuAC	NewsQA
SQuAD	0.743323	0.466377	0.494597	0.209083	0.350885
TriviaQA	0.397569	0.611938	0.444250	0.192503	0.217114
NQ	0.523388	0.463316	0.741013	0.214633	0.187447
QuAC	0.294757	0.346984	0.314440	0.341482	0.185196
NewsQA	0.373489	0.497677	0.483716	0.201628	0.530305

Όπως φαίνεται τα δύο μοντέλα που αναπτύχθηκαν με βάση τους παραπάνω συνδυασμούς παράγουν αρκετά **παρόμοια** αποτελέσματα μεταξύ τους ενώ ταυτόχρονα **προσεγγίζουν** και τα αποτελέσματα που παρουσιάζουν οι συγγραφείς. Βέβαια μπορεί να παρατηρηθεί πως η επιλογή του κατάλληλου συνδυασμού υπερπαραμέτρων **διαφέρει** από dataset σε dataset. Ένας συνδυασμός θα θεωρηθεί ως **πιο ικανοποιητικός** αν παράγει αποτελέσματα **παρόμοια ή καλύτερα** κατά την αξιολόγηση του κυρίως πάνω στο ίδιο το dataset αλλά και στα υπόλοιπα **σε σχέση με τα αποτελέσματα του paper**. Έτσι, ο συνδυασμός (1) φαίνεται να παράγει πιο ικανοποιητικά αποτελέσματα για τις περιπτώσεις του SQuAD, του TriviaQA και του QuAC dataset ενώ ο συνδυασμός (2) στα υπόλοιπα δύο datasets. Οπότε τα **τελικά αποτελέσματα** έχουν ως εξής:

	SQuAD	TriviaQA	NQ	QuAC	NewsQA
SQuAD	0.768707	0.453962	0.471520	0.208758	0.349170
TriviaQA	0.406863	0.616261	0.469505	0.198951	0.222970
NQ	0.523388	0.463316	0.741013	0.214633	0.187447
QuAC	0.309943	0.354544	0.336785	0.349126	0.210997
NewsQA	0.373489	0.497677	0.483716	0.201628	0.530305

Σύγκριση με τα αποτελέσματα του paper

SQuAD 2.0: Παράγονται αρκετά καλύτερα αποτελέσματα κατά την αξιολόγηση με τον εαυτό του και ελάχιστα καλύτερα αποτελέσματα κατά την αξιολόγηση με το QuAC. Παράγονται λίγο χειρότερα αποτελέσματα κατά την αξιολόγηση με το TriviaQA και το NQ και χειρότερα με το NewsQA.

TriviaQA: Παράγονται αρκετά καλύτερα αποτελέσματα κατά την αξιολόγηση με τον εαυτό του, με το NQ και το NewsQA. Παράγονται ελάχιστα χειρότερα αποτελέσματα κατά την αξιολόγηση με το QuAC και χειρότερα αποτελέσματα με το SQuAD 2.0.

NQ: Παράγονται αρκετά καλύτερα αποτελέσματα κατά την αξιολόγηση με τον εαυτό του, ακριβώς ίδια αποτελέσματα με το TriviaQA και αρκετά παρόμοια αποτελέσματα με το QuAC. Παράγονται λίγο χειρότερα αποτελέσματα κατά την αξιολόγηση με το SQuAD 2.0 και χειρότερα αποτελέσματα κατά την αξιολόγηση με το NewsQA.

QuAC: Παράγονται αρκετά καλύτερα αποτελέσματα κατά την αξιολόγηση με τον εαυτό του, το TriviaQA και το NewsQA και παρόμοια αποτελέσματα με το NQ. Παράγονται χειρότερα αποτελέσματα κατά την αξιολόγηση με το SQuAD 2.0

NewsQA: Παράγονται αρκετά καλύτερα αποτελέσματα κατά την αξιολόγηση με το TriviaQA και το NQ και παρόμοια αποτελέσματα με το QuAC. Παράγονται χειρότερα αποτελέσματα κατά την αξιολόγηση με τον εαυτό του και το SQuAD 2.0

Γενικό Συμπέρασμα: Όπως φαίνεται είναι αρκετά δύσκολο να επιτευχθεί μια **ισορροπία** ως προς την απόδοση ενός μοντέλου που έχει γίνει fine tuned σε ένα dataset με συγκεκριμένες υπερπαραμέτρους και τεχνικές πάνω σε όλα τα υπόλοιπα datasets. Αυτό είναι αρκετά εύκολο να το παρατηρήσει κανείς μιας και όταν **αυξάνονται** αρκετά τα f1 scores πάνω σε κάποια datasets **μειώνονται** από κάποια άλλα. Επιπλέον **σημαντικές αποκλίσεις** στα scores παρουσιάζονται τις περισσότερες φορές στο NewsQA dataset και αυτό πιθανώς να συμβαίνει στον διαφορετικό αριθμό των παρατηρήσεων που αξιοποιούν οι συγγραφείς. Ακόμα, όσον αφορά το TriviaQA, το NQ και το QuAC τα αποτελέσματα είναι πάντοτε παρόμοια ή αρκετά καλύτερα .Τέλος, όσον αφορά το SQuAD 2.0 δεν επιδέχεται πάντοτε καλύτερα αποτελέσματα όμως ο καλύτερος συνδυασμός του προηγούμενου ερωτήματος **εξακολουθεί** να παράγει καλύτερα αποτελέσματα όταν αξιολογείται με τον εαυτό του.

Επιπλέον Διευκρυνίσεις

Ερώτημα 1: Αν πρόκειται να εξεταστεί το μοντέλο στο άγνωστο test set θα πρέπει να γίνει αντικατάσταση της κλήσης model.fineTuningWithOptions με την κλήση model.fineTuning(bert_training_set, optimizer, loss_function, schelduler, max_clip_norm, epochs, batch_size) ώστε να γίνει το fine tune από την αρχή. Το όνομα του αρχείου για το ερώτημα 1 είναι ***AI2_HW4_TWITTER.ipynb***

Ερώτημα 2: Το όνομα του αρχείου για το ερώτημα 2 είναι ***AI2_HW4_SQUAD.ipynb*** και παρουσιάζεται μόνο ο καλύτερος συνδυασμός υπερπαραμέτρων.

Ερώτημα 3: Καθορίζονται κάποια flags τόσο για την επιλογή των datasets που θα γίνει το fine tuning και το evaluation όσο και για την χρήση των αποθηκευμένων μοντέλων και των datasets από το Hugging Face Hub και το Google Drive. Όλα τα flags είναι ενεργά από default (σε οποιαδήποτε περίπτωση για testing μπορούν να απενεργοποιηθούν όμως ο χώρος που δεσμεύεται και ο χρόνος εκτέλεσης θα είναι αρκετά πιο πολύς). Επιπλέον τα μοντέλα των πειραμάτων που έχουν ανέβει ακολουθούν το μοτίβο OrfeasTsk/bert-base-uncased-finetuned-<dataset>(-large-batch). Το επίθεμα (-large-batch) υποδεικνύει τον συνδυασμό υπερπαραμέτρων (2) ενώ το πρόθεμα το όνομα του προσωπικού λογαριασμού στο Hugging Face Hub. Το όνομα του αρχείου για το ερώτημα 3 είναι ***AI2_HW4_QA_DATASETS.ipynb***.

Βιβλιογραφία/Πηγές

- Διαφάνειες μαθήματος + κώδικας φροντιστηρίου
- Hugging Face documentation (tutorials + examples) & discussions
- Stack Overflow
- Stack Exchange
- GitHub discussions
- Πάρα πολλά άρθρα, blogs και forums για ιδέες πάνω στο θέμα όπως:
  https://d2l.ai/chapter_natural-language-processing-applications/finetuning-bert.html
  https://blog.ineuron.ai/A-gentle-introduction-to-BERT-Model-JfGFFXb97v
  https://skimai.com/fine-tuning-bert-for-sentiment-analysis/
  https://www.kaggle.com/akshat0007/bert-for-sequence-classification
  https://qa.fastforwardlabs.com/no%20answer/null%20threshold/bert/distilbert/exact%20match/f1/robust%20predictions/2020/06/09/Evaluating_BERT_on_SQuAD.html
  https://towardsdatascience.com/question-answering-with-a-fine-tuned-bert-bc4dafd45626
  https://medium.com/swlh/fine-tuning-bert-for-text-classification-and-question-answering-using-tensorflow-framework-4d09daeb3330
  …

Ορφέας Τσουράκης,
1115201700175