Καμάρης Άγγελος
sdi1900070

README Εργασία 4 άσκηση 2 + bonus:

Λήψη Δεδομένων:

Γι'αυτό το κομμάτι παρέχω τα ίδια τα dataframes που δημιούργησα με το αρχείο AI2_4(2)Dataframe (`my_train.csv, my_test.csv, my_val.csv`) στο οποίο δέχομαι τα δεδομένα από τα αρχεία annotated_wd_data_…_answerable.txt και κάνω split την κάθε γραμμή, αλλάζω στα relation τα R σε P (είδα στο piazza ότι έχουν το ίδιο αποτέλεσμα), εισάγω τα ids σε ένα sparql query με το οποίο κάνω το request μου και αφού πάρω τα 2 labels, φτιάχνω το span όπως είπαμε στο φροντιστήριο και το αποθηκεύω. Η όλη διαδικασία μπορεί να πάρει και 3 ώρες, αλλά δεν καταναλώνει σε μεγάλο βαθμό ram.

Περιγραφή Μοντέλου:

Το μοντέλο μου δέχεται τα tokenized inputs του Bert, το attention mask, το relation id με το όνομα label, τον δείκτη στην αρχή και τον δείκτη στο τέλος, και τον αριθμό των λέξεων στην πρόταση και χρησιμοποιεί σε όλα Linear, για να επιστρέψει πίνακες πιθανοτήτων για το relation, να είναι index στο relation_vocabulary και για το δεύτερο και τρίτο κεφάλι να είναι ένας αριθμός. Επίσης χρησιμοποιώ mask, ώστε να σιγουρευτώ ότι το ο δείκτης στο τέλος μου, δεν δείχνει έξω από την πρόταση, κάτι το οποίο μου ανέβασε το accuracy στο τέλος κοντά στο accuracy του πρώτου (πριν είχαν διαφορά 0.1 περίπου).

Preprocessing:

Δεν έκανα κάποιο ιδιαίτερο preprocessing, όμως επειδή είχα δοκιμάσει να κάνω το span όπως το έλεγε το φροντιστήριο, παρατήρησα ότι μαζί με το γεγονός ότι πρέπει να γίνει padding, οι πίνακές μου γεμίζουν μηδενικά, άρα στο τέλος το μοντέλο μαθαίνει να δημιουργεί πίνακες μόνο με 0, επομένως προτίμησα να του δίνω δείκτες στην αρχή και στο τέλος, όπως έγινε και στο paper.

Resource problem:

Το google collab παρέχει μόνο 12gb ram για χρήση το οποίο είχε ως αποτέλεσμα
να έχω κάποιους περιορισμούς στην δημιουργία του μοντέλου μου . Δηλαδή
χρησιμοποιώ μόνο 6000 γραμμές για training και 750 για test και 20 batch size..
Για να δείτε τις πλήρεις δυνατότητες του μοντέλου (εάν έχετε διαθέσιμη ram)
αφαιρέστε τις εντολές:

```
traindf=traindf.iloc[:6000, :]
testdf=testdf.iloc[:600, :]
```

Και αυξήστε το batch σύμφωνα με την ram σας. Το μεγαλύτερο που είχα καταφέρει ήταν 30, αλλά κατανάλωνε 11.5 ram, επομένως για ασφάλεια το μείωσα.

Tokenizing phase:

Το tokenizing γίνεται μέσα στο dataset που δημιούργησα, όταν καλούμε να πάρουμε το question. Ήταν πιο γρήγορο σε σχέση με το να τα κάνω ξεχωριστά και να τα κάνω και torch στο train και το test αντίστοιχα.

Fine tuning phase:

Κατά το fine tuning, χρησιμοποιώ Adamax, μιας και είναι ο Adam που μου δίνει καλύτερα αποτελέσματα σε σχέση με άλλα (μικρότερο loss κατά 0.05-0.08 αναλόγως το μοντέλο και μεγαλύτερο accuracy κατά 0.02-0.1 σε σχέση με τα άλλα).
Χρησιμοποιώ CrossEntropyLoss για την εύρεση loss, όπως είπαμε και στο φροντιστήριο και εκτυπώνω και τον αριθμό του batch στο οποίο βρισκόμαστε.

Results:

```
Relation Accuracy 0.8916666666666652, F1 score = 0.775670338090493, Precision score = 0.772235982348882, Recall score = 0.7884071826572405
Entity Start Accuracy 0.9354999999999986, F1 score = 0.8766947221519592, Precision score = 0.8819703983332956, Recall score = 0.8869151599612911
Entity End Accuracy 0.9351666666666654, F1 score = 0.8664057952359282, Precision score = 0.8708497923299506, Recall score = 0.8765658582709184
```

Relation ID:

Accuracy	F1 Score
0.8916666666666652	0.775670338090493

Entity Start:

Accuracy	F1 Score
0.9354999999999986	0.8766947221519592

Entity End:

Accuracy	F1 Score
0.9351666666666654	0.8664057952359282

Παραδοχές:

Δεδομένου ότι το μοντέλο ήθελε αρκετές ώρες να τρέξει και δεν ήταν δυνατόν να δοκιμαστούν μεγάλα νούμερα σε κάποιους παραμέτρους οι πειραματισμοί ήταν περιορισμένοι. Παρόλα αυτά παρακάτω παραθέτω κάποιους πειραματισμούς που έγιναν πάνω στο μοντέλο μου

Οι μετρήσεις που έγιναν από κάτω δοκιμάστηκαν σε 300 train και 30 test, με αρχικό batch=20 learning rate=1e-5 και dropout=0.1. Τα accuracy πάνεε με την σειρά relationid, start index, end index.

Max length:

Για max length=20 είχαμε λίγο παραπάνω από 0.08,0.58,0.58 accuracy καθώς υπήρχαν ερωτήσεις με πάνω από 20 γράμματα.
Για max length=30 είχαμε περίπου 0.1,0.62,0.57 accuracy καθώς εκεί είναι ο μέσος όρος των λέξεων στις ερωτήσεις.
Για max length=50 είχαμε περίπου 0.08,0.55,0.50 accuracy καθώς υπάρχουν αρκετές προτάσεις που αρχίζουν να δέχονται padding.
Για max length=100 έχουμε 0.07,0.54,0.51 accuracy αλλά η διαφορά είναι εμφανής και στο loss, καθώς έχει ανεβεί κατά 0.2 σε σχέση με το προηγούμενο.
Και γενικώς όσο μεγαλύτερο γινόταν ύστερα, τόσο χειρότερο accuracy είχε, τόσο περισσότερη ram κατανάλωνε και τόσο περισσότερο χρόνο ήθελε.
Επέλεξα να κρατήσω το max length=30, μιας και δίνει τα καλύτερα αποτελέσματα και καταναλώνει λίγη ram και λιγότερο χρόνο.

Epochs
Οι μόνες χρονικά λογικές επιλογές ήταν 1 εως 5 epoch . Κατέληξα στα 2 γιατί αν και θέλει σχεδόν διπλάσιο χρόνο σε σχέση με το 1 βελτιώνει αρκετά την αποτελεσματικότητα του μοντέλου. Φυσικά αν κάποιος είχε αρκετό χρόνο θα μπορούσε να βάλει και 4 ή 5 αλλά εκεί υπάρχει ρίσκο για overfitting.

Batch size
Το batch size έχει ίσως και την μεγαλύτερη επιρροή στην ram, καθώς το μοντέλο μου δεν μπορεί να τρέξει για batch μεγαλύτερο του 30. Δεν επηρέασε σε μεγάλο βαθμό την απόδοση, αν και για batches από 15 έως το 30 παρατηρήθηκε αύξηση του accuracy σε σχέση με τα 5 και 10 κατά 0.01 εώς 0.1. Κράτησα το batch size στο 20, μιας και ήταν η πιο ασφαλής επιλογή από άποψη ram.

Dropout
Το Dropout επηρεάζει αρκετά το accuracy, καθώς για dropout=0.5 και 0.05 είχαμε accuracy=0.01,0.03,0.02(0.018 για το 0.05) ενώ για dropout=0.01 ανέβηκαν τα προηγούμενα accuracy, 0.02, 0.015 και 0.005 αντιστοίχως.
Επομένως έμεινα στο 0.1 που όπως φαίνεται και από τα προηγούμενα αποτελέσματα έχει την μεγαλύτερη ακρίβεια.

Learning rate
Το Learning rate είχε τον αισθητά μεγαλύτερο αντίκτυπο σε σχέση με όλα τα υπόλοιπα, στο accuracy, καθώς αν και το 1e-5, 1e-6, 1e-3 είχαν πάνω κάτω τα αποτελέσματα που φάνηκαν από επάνω, με διαφορά 0.02-0.05 πόντους πάνω ή κατω σε κάθε head, το 1e-4 έκανε την αισθητά μεγαλύτερη διαφορά, με 0.1-0.2 πόντους αύξηση σε κάθε head.

Bonus:

Επειδή δεν έχω προπονήσει το μοντέλο μου σε πολύ μεγάλο όγκο δεδομένων, ακολουθόντας τις οδηγίες του φροντιστηρίου καθώς και τις σημειώσεις από το paper, έφτιαξα το bonus, το οποίο όμως μπορεί να μην βρίσκει πάντα απάντηση. Δεν δημιούργησα την λίστα από δείκτες όπως λέει το φροντιστήριο, ο λόγος θα φανεί πιο κάτω που εξηγώ τον κώδικά μου.

Το μπόνους βρίσκεται στο κάτω μέρος του colab μου για αυτή την εργασία. Δέχεται μια ερώτηση, την περνάει στο μοντέλο μου το μοντέλο επιστρέφει τους τρεις πίνακες, Από τους οποίους κρατάμε την μεγαλύτερη πιθανότητα, για το relation_id, χρησιμοποιούμε την πρόβλεψη σαν index και παίρνουμε το αποτέλεσμα στο relation vocabulary, ενώ για το entity label, κρατάω σε ένα string τις λέξεις από εκεί που δείχνει η αρχή έως το τέλος στην ερώτηση, χρησιμοποιώ fuzzywuzzy, όπως και στο paper, για να βρω το πιο κοντινό entity label από αυτά που έδωσα να προπονηθεί και να κριθεί το μοντέλο μου, μετά βρίσκω το id τους κρατόντας το index της τοποθεσίας αυτής της λέξης, και τέλος στέλνω ένα api όπως στο παράδειγμα του φροντιστηρίου με το relation id και το entity id που βρήκα.(Σε περίπτωση που δεν βρεθεί απάντηση, εκτυπώνω ότι δεν βρήκα απάντηση.)