Angelos Kamaris sdi1900070
AI 2 - project 2

December 23, 2022

Καμάρης Άγγελος sdi1900070

1 Πηγές

Αξιοποίησα τον κώδικα του φροντιστηρίου κατά κύριο λόγο, χρησιμοποιώντας επίσης εντολές από το sklearn. Αξιοποίησα τον οδηγό:
https://towardsdatascience.com/pytorch-tabular-binary-classification-a0368da5bb89
για την δημιουργία του νευρωνικού δικτύου μου καθώς επίσης αξιοποίησα σαν σκελετό την προηγούμενη εργασία μου, κρατόντας το preprocessing των δεδομένων μου καθώς και τον χωρισμό των δεδομένων, την εισαγωγή τεστ και την εμφάνιση αποτελεσμάτων.

2 Επεξήγηση Κώδικα

Χρησιμοποίησα τις στήλες rating και review σαν Υ και X αντίστοιχα, όπου επεξεργάστηκα τα rating, έτσι ώστε τα "κακά" (0-4) να έχουν την τιμή 0 και τα "καλά" (7-10) να έχουν την τιμή 1.

Επεξεργάστηκα τα δεδομένα από τα reviews κάνοντας τα κεφαλαία μικρά, αυτή την φορά κράτησα τα σημεία στίξης, έβγαλα τις λέξεις που είχαν λιγότερα από 2 γράμματα, βρίσκοντας τις πιο πολυχρησιμοποιημένες λέξεις και αφαιρώντας τις πιο σπάνιες, και τέλος έκανα lematize , χωρίς stematize φυσικά για να αναγνωρίζονται οι λέξεις.

Χώρισα τα δεδομένα μου έτσι ώστε 30% εξ αυτών να γίνονται validate και τα υπόλοιπα να χρησιμοποιούνται για το training και για την εισαγωγή των δεδομένων στο νευρονικό δίκτυο, τα πέρασα από το glove2word2vec βάζοντας σαν input το glove.6B.300d.txt καθώς με αυτό έχω την μεγαλύτερη ακρίβεια (από 76% σε 84%).

Χρησιμοποιώ Adadelta για να κάνω optimization καθώς μου έδωσε τα καλύτερα αποτελέσματα. Χρησιμοποιώ learning rate=0.001, batch size =20 , number of

epochs=100και το νευρωνικό μου δίκτυο έχει: h1=300, h2=150, h3=75. Αξιοποιώ το learning curvetης πρώτης εργασίας για να εκτυπώσω τις αποδόσεις του νευρωνικού δικτίου μου με τον αριθμό των δεδομένων, σύμφωνα με το F1-score τους και το μέγεθος των δεδομένων και εμφανίζω το losses plot. Τέλος εκτυπώνω τα: F1-Score, Recall, Precision για τα training και test που χώρισα πριν καθώς και το ROC plot .

Όλες οι αλλαγές στις οποίες αναφέρθηκα από πάνω αποσκοπούν στην εύρεση του καλύτερου αποτελέσματος. Ύστερα από δοκιμές είδα ότι χωρίς να κάνω overfitting καταφέρνω να βρίσκω μεγαλύτερο validation από train , και καταλήγω με loss=2.63 που είναι το χαμηλότερο που μπορώ να φτάσω σε λογικό χρόνο, χωρίς να χαλάω το recall .
Για την εισαγωγή ενός test, αρκεί να εισάγετε τα δεδομένα στην μεταβλητή test df , όπως το παράδειγμα στα σχόλια.

3 Παρατηρήσεις

Την μεγαλύτερη αλλαγή την παρατήρησα, στην χρήση preprocessing, όταν το μοντέλο δεν έκανε overfit ή underfit καθώς παρατηρούνται αλλαγές της τάξης $0.06 - 0.08$, αναλόγως και τα υπόλοιπα δεδομένα. Συγκεκριμένα τα σημεία στίξης φαίνεται να βοηθάνε το μοντέλο καθώς επίσης και όσο μεγαλύτερος ο πίνακας από το glove τόσο μεγαλύτερο το accuracy.

Το batch size φαίνεται να επηρεάζει σε μεγάλο βαθμό το loss , καθώς όσο μικρότερο είναι τόσο λιγότερο θα χάνουμε αλλά έτσι μειώνεται και το recall . Τα layers φαίνεται να επηρεάζουν επίσης πολύ τα αποτελέσματα, καθώς για πολύ μικρά, δεν έχουμε καλή ακρίβεια αλλά για πολύ μεγάλα το πρόγραμμα γίνεται αργό και μπορεί πάλι να καταλήξουμε με κακή ακρίβεια. Τέλος το learning rate καθώς και ο optimizator φαίνεται να έχουν την δυνατότητα να αυξήσουν την ακρίβεια καθώς και να μειώσουν το loss ή να χαλάσουν την ακρίβεια.

4 Αποτελέσματα

Τα αποτελέσματα που έχει το πρόγραμμά μου για τα δεδομένα που ανέφερα είναι:
F1 Score (train): 0.8423349699945445 F1 Score (validation): 0.8367087661609746
Recall Score (train): 0.8336403480912151 Recall Score (validation): 0.8274944567627495
Precision Score (train): 0.8512128680762745 Precision Score (validation): 0.846130592503023