Νευρο-Ασαφής Υπολογιστική
CODING PROJECT

Γιοβανούδη Ελένη 2509, Κυριακού Γεωργία 2901

February 2022

1 Εισαγωγή

Η συγκεκριμένη εργασία πραγματεύεται την ανάπτυξη ενός νευρωνικού δικτύου κατηγοριοποιητή, που διακρίνει άρθρα σε ψευδή και αληθή. Παρακάτω αναλύουμε τη διαδικασία κατασκευής του δικτύου, τα μέρη που το συνιστούν καθώς και τα αποτελέσματά του.

2 Προετοιμασία Δεδομένων Εισόδου

Μας δίνεται ένα αρχείο τύπου json που περιλαμβάνει συνδέσμους προς τα διάφορα άρθρα με τα οποία θα εκπαιδεύσουμε το νευρωνικό δίκτυο. Τα αρχεία αυτά εισέρχονται σε ένα αρχείο csv, του οποίου η κάθε γραμμή περιλαμβάνει τον τίτλο, το κείμενο και το overall rating του εκάστοτε άρθρου. Στη συνέχεια, ο τίτλος ενώνεται με το υπόλοιπο κείμενο, ενώ το label μετατρέπεται σε 0, εάν το overall rating κυμαίνεται από 0 έως 0.4, ή σε 1, εάν κυμαίνεται από 0.6 μέχρι 1. Στην περίπτωση που είναι 0.5, το κείμενο παραλείπεται.

Προκειμένου να εξάγουμε τα αποτελέσματά μας, χρειάζεται επεξεργασία των κειμένων που διακρίνεται σε τρία βηματα:

1. Αφαίρεση ανεπιθύμητων χαρακτήρων, όπως σημεία στίξης και ειδικοί χαρακτήρες.
2. Αφαίρεση των λεγόμενων stop words, δηλαδή λέξεις που δεν έχουν εννοιολογική σημασία.
3. Μετατροπή των λέξεων στη λεξικογραφική τους μορφή.

Στη συνέχεια, για κάθε κείμενο δημιουργείται ένα διάνυσμα όπου κάθε κόμβος αντιστοιχεί σε μια λέξη του κειμένου και έπειτα σε κάθε λέξη ανατίθεται ένας μοναδικός ακέραιος, όμως οι ίδιες λέξεις έχουν τον ίδιο αριθμό, μέσω της διαδικασίας Tokenization.

3 Αρχιτεκτονική του νευρωνικού δικτύου

Το πρώτο βήμα στην αρχιτεκτονική που επιλέξαμε είναι το word embedding, κατά το οποίο δημιουργούνται διανύσματα που παρουσιάζουν την έννοια κάθε λέξης ανάλογα με την πρόταση στην οποία ανήκουν. Συγκεκριμένα, για κάθε λέξη, το μήκος του διανύσματος είναι 300 και συλλέγονται τα διανύσματα από τις πρώτες 512 λέξεις του κάθε άρθρου.

Έπειτα χρησιμοποιούμε δύο convolutional layers, το πρώτο με 64 φίλτρα και το δεύτερο με 128, με μέγεθος kernel 5, ακολουθούμενα το καθένα από ένα Max Pooling layer.

Μετά υλοποιούμε ένα Recurrent Neural Network, συγκεκριμένα LSTM, με 32 νευρώνες, ακολουθούμενο από ένα dropout layer με 25 % droupout rate.

Τέλος προσθέτουμε ένα fully-connected layer με sigmoid activation function και δίνει ως αποτέλεσμα 0 ή 1.

Παρακάτω παρουσιάζεται το μοντέλο όπως εκτυπώνεται από τον κώδικα:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_32 (Embedding)    (None, 512, 300)          4891200

 conv1d_90 (Conv1D)          (None, 512, 64)           96064

 max_pooling1d_90 (MaxPoolin  (None, 256, 64)          0
 g1D)

 conv1d_91 (Conv1D)          (None, 256, 128)          41088

 max_pooling1d_91 (MaxPoolin  (None, 128, 128)         0
 g1D)

 lstm_57 (LSTM)              (None, 32)                20608

 dropout_62 (Dropout)        (None, 32)                0

 dense_32 (Dense)            (None, 1)                 33

=================================================================
Total params: 5,048,993
Trainable params: 5,048,993
Non-trainable params: 0
_________________________________________________________________
```

4 Εκπαίδευση του νευρωνικού δικτύου

Τα δεδομένα που δίδονται χωρίζονται σε 0.3 για testing 0.7 για training και στην συνέχεια τα training δεδομένα χωρίζονται σε 0.2 για validation και 0.8 για training. Το δίκτυο λαμβάνει ως είσοδο τα δεδομένα και κάνει 200 επαναλήψεις το μέγιστο. Το batch size είναι 256 και ως optimizer χρησιμοποιούμε τον αλγόριθμο Adam. Υλοποιούμε, επίσης, τον μηχανισμό early stopping, κατά τον οποίο αν το validation accuracy παραμένει ίδιο για 30 επαναλήψεις, το πρόγραμμα σταματά πρόωρα καθώς το νευρωνικό δίκτυο συγκλίνει. Το μοντέλο, ακόμη, κάθε φορά που βρίσκει ένα βελτιωμένο validation accuracy, αποθηκεύει τα νέα βάρη σε ένα αρχείο και έχει ως αποτέλεσμα, ακόμη κι αν στις επόμενες επαναλήψεις μειωθεί το accuracy στο τέλος να έχουμε κρατήσει το καλύτερο.

5 Εκπαιδευμένες τιμές των παραμέτρων και χρόνοι απόκρισης

Ο χρόνος απόκρισης του δικτύου όσον αφορά την εκπαίδευση του είναι περίπου 100 δευτερόλεπτα και όσον αφορά το testing του δικτύου είναι 0.7 δευτερολέπτα.Επίσης,κάθε επανάληψη είναι περίπου 2 δευτερόλεπτα.Όσον αφορά τις τελικές τιμές των παραμέτρων του αυτές μπορούν να ληψθούν με το αρχείο get_weights και δε θα αναφερθούν εδώ διότι είναι πάρα πολλές.Ωστόσο,αν κάποιος θέλει να φορτώσει το τελικό μοντέλο μπορεί με την συνάρτηση load_model του keras (παράδειγμα υπάρχει στο get_weights).

	precision	recall	f1-score	support
0	0.62	0.33	0.43	24
1	0.71	0.89	0.79	44
accuracy			0.69	68
macro avg	0.66	0.61	0.61	68
weighted avg	0.68	0.69	0.66	68

6 Αποτίμηση της επίδοσης με διαφορετικά δεδομένα εισόδου

Όπως αναφέρθηκε παραπάνω χωρίζουμε τα δεδομένα για training,testing Και validation.Η επίδοση του δικτύου πάνω σε αυτά τα δεδομένα κυμαίνεται περίπου στο 0.7.Δεν υπάρχει πάντα σταθέρο νούμερο διότι τα δεδομένα είναι αρκέτα λιγά και ανάλογα με τον διαχωρισμό τους παράγονται άλλες επιδόσεις.

7 Έλεγχος της επίδοσης του δικτύου με δικά σας δεδομένα

Για να γίνει πιο εύκολο το testing του δικτύου με δικά σας δεδομένα δημιουργήσαμε έναν wrapper στο αρχείο wrapper.py.Στο συγκεκριμένο αρχείο αρκεί απλά κάποιος να εισάγει το όνομα του .csv που θέλει να φορτώσει δεδομένα,το όνομα της στήλης που περιέχουν τα αρχεία,το label και τους τιτλούς(εάν υπάρχουν).Στο αρχείο δίνονται περισσότερες πληροφορίες για το πώς καλείται ο wrapper/function.

8 Συμπεράσματα

Μετά από πολλές δοκιμές διαφόρων μοντέλων, εισαγωγή layer και ρύθμιση τιμών των παραμέτρων, καταλήγουμε στο παραπάνω μοντέλο. Το περιορισμένο dataset που μας δίνεται και η ύπαρξη εσφαλμένων συνδέσμων έχει ως αποτέλεσμα να μην έχουμε ξεκάθαρη εικόνα για την βέλτιστη αρχιτεκτονική που θα μπορούσαμε να υλοποιήσουμε καθώς δοκιμάζοντας διάφορες παραλλαγές, δεν παρατηρήσαμε κάποια βελτίωση. Ωστόσο, στην περίπτωση που δεν αφαιρούσαμε τα stop words από το κείμενο, η βελτίωση ήταν εμφανής, αλλά για λόγους ορθότητας του μοντέλου, επιλέξαμε να το συμπεριλάβουμε σαν τεχνική.

9 Βιβλιογραφία

1. Divya Raghunathan, 2020, https://towardsdatascience.com/nlp-in-python-data-cleaning-6313a404a470
2. Chirag Goyal, 2021, https://www.analyticsvidhya.com/blog/2021/06/part-3-step-by-step-guide-to-nlp-text-cleaning-and-preprocessing/?fbclid=IwAR3WX6d8DB4m5V-NLLdQ-mkvGjqTCfimf0iCUGiMydrtfmzyENHP5IsrKzs
3. Jamal AbdulNasir, Osama SubhaniKhan, Iraklis Varlamis, 2021, https://www.sciencedirect.com/science/article/pii/S2667096820300070