ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία Φωνής και Φυσικής Γλώσσας
Εαρινό Εξάμηνο 2022-23

3η Εργαστηριακή Άσκηση

1 Εισαγωγή

Στην προπαρασκευή, σχεδιάσατε απλές αρχιτεκτονικές νευρωνικών δικτύων για κατηγοριοποίηση κειμένων, χρησιμοποιώντας προ-εκπαιδευμένες διανυσματικές αναπαραστάσεις λέξεων (pretrained word embeddings). Συγκεκριμένα, η αρχιτεκτονική η οποία σας έχει ζητηθεί να φτιάξετε, όπως φαίνεται στην Εικόνα 1, είναι η εξής:

$$e_i = embed(x_i) \qquad \text{διανυσματική αναπαράσταση κάθε λέξης (embedding)} \qquad (1)$$

$$u = \frac{1}{N}\sum_{i=1}^{N} e_i \qquad \text{αναπαράσταση κειμένου: μέσος όρος embeddings} \qquad (2)$$

$$r = ReLU(Wu + b) \qquad \text{μη-γραμμικός μετασχηματισμός} \qquad (3)$$

όπου r η διανυσματική αναπαράσταση ενός κειμένου (feature vector).
Σκοπός αυτής της εργαστηριακής άσκησης είναι να δημιουργήσετε καλύτερες αναπαραστάσεις:

1. συνενώνοντας τις αναπαραστάσεις που προκύπτουν μέσω του μέσου όρου (mean pooling) και του μεγίστου ανά διάσταση (max pooling) των word embeddings κάθε πρότασης
2. χρησιμοποιώντας Ανατροφοδοτούμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks) και συγκεκριμένα LSTMs (Long Short-Term Memory networks)
3. εφαρμόζοντας απλό μηχανισμό προσοχής (Self-Attention) επί των word embeddings
4. εφαρμόζοντας πολλαπλό μηχανισμό προσοχής (MultiHead-Attention) επί των word embeddings
5. αναπτύσσοντας και εφαρμόζοντας το κομμάτι του Encoder ενός Transformer (Transformer-Encoder) επί των word embeddings
6. χρησιμοποιώντας διαθέσιμα προ-εκπαιδευμένα μοντέλα (Pre-Trained Transformers) για την κατηγοριοποίηση του συναισθήματος
7. εκπαιδεύοντας/κουρδίζοντας (fine-tuning) προ-εκπαιδευμένα μοντέλα για την κατηγοριοποίηση του συναισθήματος

2 Θεωρητικό Υπόβαθρο

2.1 Ανατροφοδοτούμενα Νευρωνικά Δίκτυα

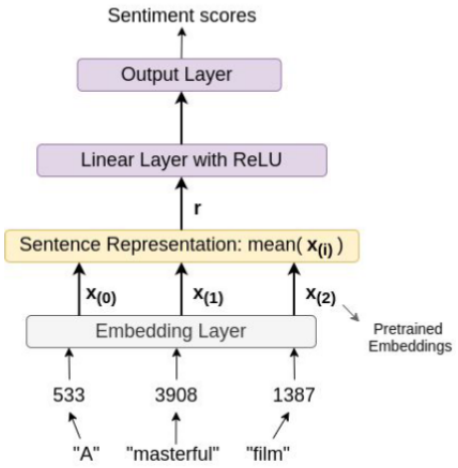
Τα Ανατροφοδοτούμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNN) σε αντίθεση με άλλες αρχιτεκτονικές έχουν την ιδιότητα ότι σχηματίζουν συνδέσεις με ανάδραση (feedback). Τα περισσότερα δίκτυα απαιτούν εισόδους σταθερών διαστάσεων ενώ τα RNNs μπορούν να επεξεργάζονται με ευκολία δεδομένα μεταβλητού μήκους.

Αυτή η ευελιξία, τα κάνει ιδανικά για την επεξεργασία ακολουθιών, όπως σε προβλήματα επεξεργασίας φυσικής γλώσσας. Ο τρόπος λειτουργίας τους μοιάζει με τον τρόπο που ο άνθρωπος επεξεργάζεται την γλώσσα (κείμενο, ομιλία), δηλαδή σειριακά.

Η βασική λειτουργία ενός RNN είναι η εξής: δέχεται ως είσοδο μία ακολουθία από διανύσματα $x = (x_1, x_2, ..., x_n)$ και παράγει ως έξοδο ένα μοναδικό διάνυσμα y. Όμως, η κρίσιμη διαφορά είναι ότι σε κάθε βήμα, για την παραγωγή του αποτελέσματος, λαμβάνεται υπόψη το αποτέλεσμα του προηγούμενου βήματος. Το απλό RNN, το οποίο εξετάζουμε τώρα, έχει ορισμένες παραλλαγές [Hopfield, 1982, Elman, 1990, Jordan, 1997]. Η παραλλαγή η οποία εξετάζουμε εδώ είναι το δίκτυο Elman [Elman, 1990].

Η Εικόνα 2 ξεκαθαρίζει τον τρόπο λειτουργίας ενός RNN. Όπως φαίνεται και από το σχήμα, ένα RNN μπορεί να δεχθεί μία ακολουθία οποιουδήποτε μήκους. Αφού επεξεργαστεί ένα-προς-ένα τα στοιχεία, παράγει το τελικό αποτέλεσμα y_n, το οποίο είναι μία *σταθερή διανυσματική αναπαράσταση* για όλη την ακολουθία.

Figure 1: Συνολική αρχιτεκτονική Προπαρασκευής



Παράδειγμα Ας δούμε ένα συνηθισμένο παράδειγμα εφαρμογής ενός RNN στην επεξεργασία φυσικής γλώσσας. Σε ένα πρόβλημα κατηγοριοποίησης κειμένου, το RNN επεξεργάζεται τις λέξεις του εγγράφου, τη μία μετά την άλλη, και στο τέλος παράγει την *διανυσματική αναπαράσταση του εγγράφου* y_n. Η αναπαράσταση αυτή χρησιμοποιείται σαν διάνυσμα χαρακτηριστικών για την κατηγοριοποίηση του κειμένου, π.χ. βάση του συναισθηματικού του προσανατολισμού.

Πιο συγκεκριμένα, το έγγραφο αναπαριστάται από μία ακολουθία λέξεων. Κάθε λέξη αναπαριστάται από ένα διάνυσμα (word embedding) x_i, με $x_i \in R^E$, όπου E οι διαστάσεις των διανυσμάτων λέξεων. Έτσι έχουμε την ακολουθία $X = (x_1, x_2, ..., x_T)$, όπου T το πλήθος των λέξεων στο έγγραφο. Το RNN επεξεργάζεται σειριακά τις λέξεις, διατηρώντας στο εσωτερικό του, μία περίληψη όσων έχει διαβάσει μέχρι τη χρονική στιγμή t. Στο τέλος, περιέχει την περίληψη όλης της πληροφορίας του εγγράφου και από αυτή αποτελεί την τελική διανυσματική αναπαράσταση για το έγγραφο.

2.1.1 Αμφίδρομο RNN

Ένα αμφίδρομο RNN (bidirectional RNN ή BiRNN) αποτελείται από τον συνδυασμό δύο διαφορετικών RNN, όπου το κάθε ένα επεξεργάζεται την ακολουθία με διαφορετική φορά. Το κίνητρο αυτής της τεχνικής, είναι η δημιουργία μίας σύνοψης του εγγράφου και από τις δύο κατευθύνσεις, ώστε να σχηματιστεί μία καλύτερη αναπαράσταση. Έτσι έχουμε ένα δεξιόστροφο RNN $\overrightarrow{f}$, το οποίο διαβάζει μία πρόταση από το x_1 προς x_T και ένα αριστερόστροφο RNN $\overleftarrow{f}$, το οποίο διαβάζει μία πρόταση από το x_T προς x_1. Έτσι, κάθε χρονική στιγμή t, έχουμε:

$$h_i = \overrightarrow{h_i} \parallel \overleftarrow{h_i}, \quad h_i \in R^{2N} \tag{4}$$

όπου το $\parallel$ συμβολίζει την πράξη της ένωσης δύο διανυσμάτων και N είναι οι διαστάσεις του κάθε RNN.

2.1.2 Βαθιά RNN

Όπως και με τα απλά FFNN μπορούμε να στοιχίσουμε ένα RNN σε επίπεδα για την δημιουργία βαθιών δικτύων.

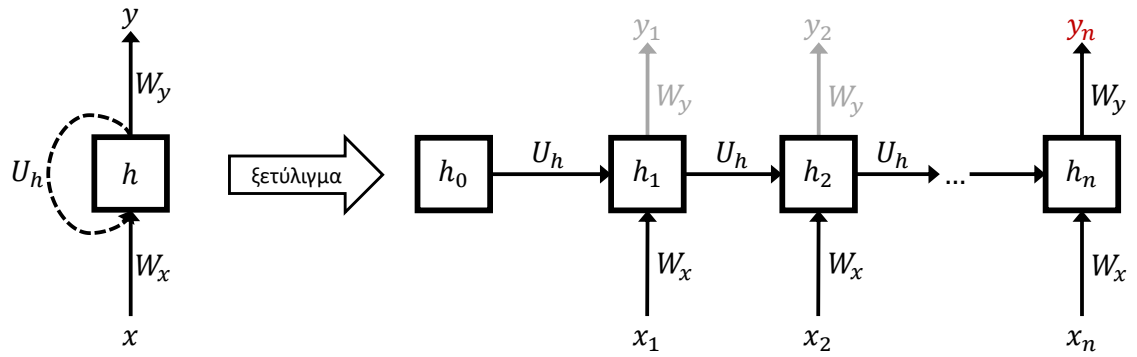


Figure 2: Διάγραμμα λειτουργίας ενός RNN. Υπάρχουν δύο τρόποι ώστε να σκέφτεται κανείς το πως λειτουργεί ένα RNN. Στην αριστερή εικόνα φαίνεται η αναδρομική λειτουργία του RNN. Το RNN δέχεται το ένα μετά το άλλο τα στοιχεία της ακολουθίας και ενημερώνει την εσωτερική ή κρυφή του κατάσταση. Ένας άλλος τρόπος αναπαράστασης είναι με το "ξετύλιγμα" του δικτύου στο χρόνο. Ουσιαστικά ένα RNN είναι ένα feed-forward NN (FFNN) με ανάδραση. Στην ξετυλιγμένη μορφή, ένα RNN μοιάζει με ένα πολυεπίπεδο FFNN.

2.1.3 Long Short-Term Memory (LSTM)

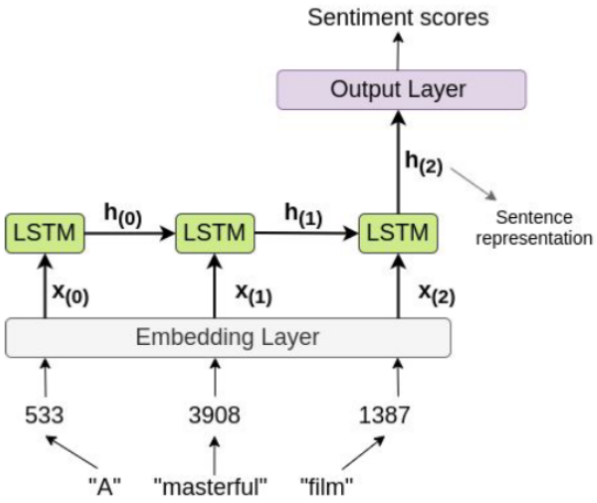
Πλέον δεν χρησιμοποιείται σχεδόν καθόλου το απλό RNN αλλά παραλλαγές του, οι οποίες επιχειρούν να ξεπεράσουν ορισμένα προβλήματα, με σημαντικότερο αυτό του vanishing gradient κατά την διάρκεια της εκπαίδευσης του δικτύου [Bengio et al., 1994,Hochreiter et al., 2001]. Η πιο δημοφιλής παραλλαγή είναι το δίκτυο Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997].

Χρησιμοποιεί έναν εξεζητημένο μηχανισμό, ο οποίος του επιτρέπει να ξεπεράσει το πρόβλημα του RNN, σχετικά με την αναγνώριση απομακρυσμένων εξαρτήσεων.

Το LSTM έχει δύο βασικές διαφορές από το απλό RNN:

- *Δεν* εφαρμόζει συνάρτηση ενεργοποίησης στις αναδρομικές συνδέσεις. Αυτό σημαίνει ότι οι ενημερώσεις θα είναι γραμμικές. Έτσι εγγυάται ότι τα σφάλματα (gradients), δεν θα εξαφανίζονται από την επαναληπτική εφαρμογή των ενημερώσεων (backpropagation through-time). Συνεπώς εξασφαλίζει τη ροή της πληροφορίας στο δίκτυο.
- Μηχανισμός με θύρες. Ο μηχανισμός αυτός εισάγει θύρες, οι οποίες ρυθμίζουν το πόσο θα ενημερώνεται κάθε διάνυσμα του δικτύου (εσωτερική κατάσταση, έξοδος κλπ.). Με αυτό τον τρόπο το δίκτυο αφομοιώνει και διατηρεί τις πιο σημαντικές πληροφορίες καλύτερα.

Figure 3: Συνολική αρχιτεκτονική με χρήση LSTM



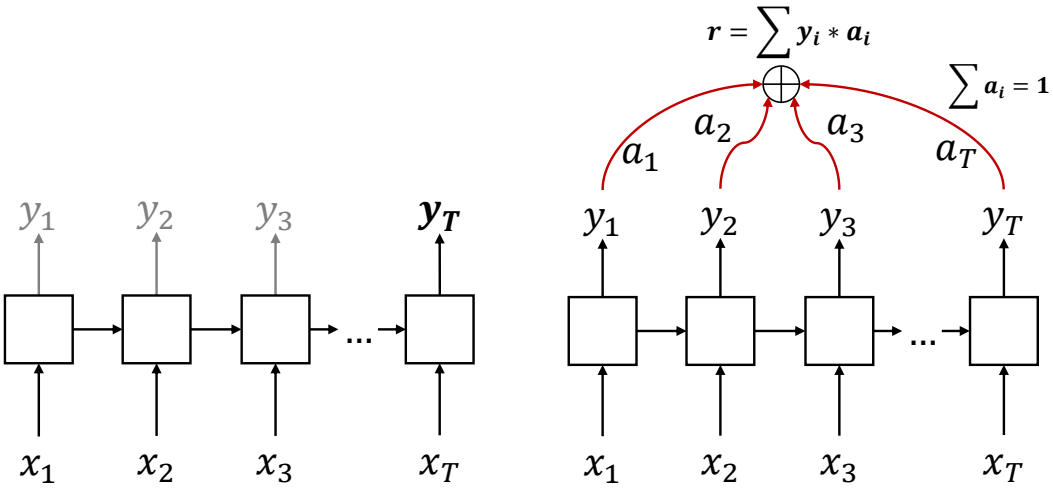
Περισσότερα στοιχεία για τη λειτουργία του LSTM μπορείτε να βρείτε εδώ[1]. Στην Εικόνα 3

[1] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

παρουσιάζεται η συνολική αρχιτεκτονική που θα προκύψει από τη χρησιμοποίηση του LSTM για την κατηγοριοποίηση του συναοσθήματος.

2.2 Μηχανισμός Προσοχής (Attention)

Ο μηχανισμός προσοχής άλλαξε ριζικά το πεδίο της επεξεργασίας φυσικής γλώσσας, λόγω της ικανότητάς να αναγνωρίζει πληροφορία σε μια είσοδο που είναι η καταλληλότερη για την επίτευξη κάποιας εργασίας. Πρατάθηκε από τους [Bahdanau et al., 2014] στο άρθρο με τίτλο"Neural Machine Translation by Jointly Learning to Align and Translate" ως ένας μηχανισμός που επέτρεπε στον decoder να εστιάσει στις καταλληλότερες λέξεις σε κάθε βήμα. Χρησιμοποιώντας ένα RNN για την κωδικοποίηση των βημάτων της ακολουθίας, πέτυχε τη μείωση της απόστασης μεταξύ δύο "απομακρυσμένων" αλλά σχετικών λέξεων της ακολουθίας, με το μηχανισμό που φαίνεται στην Εικόνα 4.



(a) Απλό RNN. Η τελική αναπαράσταση της ακολουθίας είναι η τελευταία έξοδος του RNN y_n.

(b) RNN με μηχανισμό attention. Η τελική αναπαράσταση της ακολουθίας είναι το σταθμισμένο άθροισμα των ενδιάμεσων εξόδων y_i.

Figure 4: Σύγκριση μεταξύ του απλού RNN και ενός RNN με attention. Στο RNN με attention, Η τελική αναπαράσταση r είναι το σταθμισμένο άθροισμα όλων των εξόδων του RNN. Τα βάρη κάθε βήματος a_i, ορίζονται από το attention layer.

2.3 Transformer

Σε ένα πρωτοποριακό άρθρο οι συγγραφείς [Vaswani et al., 2017] πρότειναν πως "Attention Is All You Need". Κατάφεραν να δημιουργήσουν μια αρχιτεκτονική που ονόμασαν Transformer, η οποία βελτίωσε κατά πολύ τα αποτελέσματα στη μετάφραση κειμένου χωρίς να χρησιμοποιούν συνελικτικά ή αναδρομικά δίκτυα, παρά μόνο μηχανισμούς προσοχής (συν κάποια άλλα στοιχεία όπως στρώματα κανονικοποίησης κ.α.). Η αρχιτεκτόνικη αυτή φαίνεται στην Εικόνα 5 μαζί με κάποια σχόλια για τις επιμέρους μονάδες της. Περισσότερες πληροφορίες για τη λειτουργία του Transformer μπορεί κανείς να βρει εδώ[2].

Ένα πολύ λεπτομερές tutorial από τον Andrej Karpathy με τον τίτλο "Let's build GPT: from scratch, in code, spelled out" μπορεί κανείς να παρακολουθήσει εδώ[3], στο οποίο ξεκαθαρίζονται πολλές έννοιες σχετικά με τους Transformers. Διαλέγοντας κάποιες σημειώσεις του A. Karpathy από το παραπάνω:

[2] https://jalammar.github.io/illustrated-transformer/
[3] https://youtu.be/kCc8FmEb1nY

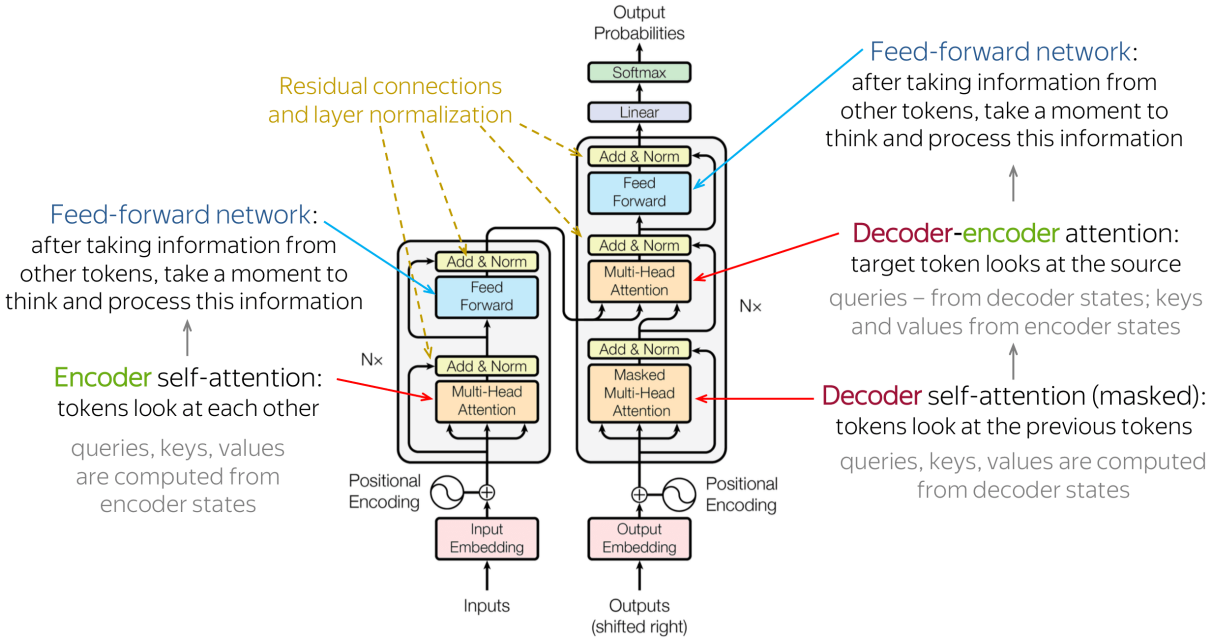


Figure 5: Αρχιτεκτονική Transformer

- ο μηχανισμός προσοχής (Attention) είναι ένας μηχανισμός επικοινωνίας. Μπορεί κανείς να τον δει σαν κόμβους ενός κατευθυνόμενου γράφου που συνδέονται μεταξύ τους και η πληροφορία συγκεντρώνεται με ένα σταθμισμένο άθροισμα από όλους τους κόμβους που δείχνουν κάθε κόμβο με βάρη που εξαρτώνται από δεδομένα.

- Δεν υπάρχει η έννοια χώρου. Ο μηχανισμός προσοχής απλώς δρα σε ένα σύνολο διανυσμάτων. Αυτός είναι ο λόγος για τον οποίο προσθέτουμε μια κωδικοποίηση θέσης.

- Ο μηχανισμός Self-attention σημαίνει απλώς ότι τα κλειδιά (keys) και οι τιμές (values) παράγονται από την ίδια πηγή με τα ερωτήματα (queries). Στον "cross-attention" μηχανισμό, τα ερωτήματα παράγονται από μία πηγή αλλά τα κλειδιά και οι τιμές προέρχονται από κάποια άλλη (π.χ. έναν encoder).

2.4 Pre-Trained Transformers

Τα τελευταία χρόνια, αρκετά προ-εκπαιδευμένα μοντέλα γίνονται διαθέσιμα σε πλατφόρμες όπως το HuggingFace[4]. Με την κατάλληλη αξιοποίηση πακέτων όπως το transformers[5], κάποιος μπορεί να χρησιμοποιήσει χιλιάδες προεκπαιδευμένα μοντέλα για την εκτέλεση εργασιών σε κείμενο, εικόνα και ήχο.

Ενδεικτικά, τα μοντέλα μπορούν να εφαρμοστούν σε:

- Κείμενο, για εργασίες όπως ταξινόμηση κειμένου, εξαγωγή πληροφοριών, απάντηση ερωτήσεων, σύνοψη, μετάφραση, δημιουργία κειμένου, σε περισσότερες από 100 γλώσσες.

- Εικόνες, για εργασίες όπως ταξινόμηση εικόνων, ανίχνευση αντικειμένων και τμηματοποίηση.

- Ήχο, για εργασίες όπως η αναγνώριση ομιλίας και η ταξινόμηση ήχου.

[4] https://huggingface.co/
[5] https://github.com/huggingface/transformers

3 Ερωτήματα

Για την υλοποίησή σας, βασιστείτε στον διαθέσιμο κώδικα που βρίσκεται εδώ[6].

Ερώτημα 1
1.1 Υπολογίστε την αναπαράσταση κάθε πρότασης u (Εξίσωση 2) ως την συνένωση (concatenation) του μέσου όρου (mean pooling) και του μεγίστου ανά διάσταση (max pooling) των word embeddings κάθε πρότασης, $E = (e_1, e_2, ..., e_N)$.

$$u = [mean(E)||max(E)] \tag{5}$$

1.2 Τι διαφορά(ές) έχει αυτή η αναπαράσταση με την αρχική; Τι παραπάνω πληροφορία θα μπορούσε να εξάγει; Απαντήστε συνοπτικά.

Ερώτημα 2
Σε αυτό το ερώτημα θα πρέπει να χρησιμοποιήσετε ένα LSTM για να κωδικοποιήσετε την πρόταση. Το LSTM θα διαβάζει τα word embeddings e_i και θα παράγει μία νέα αναπαράσταση για κάθε λέξη h_i, η οποία θα λαμβάνει υπόψη της και τα συμφραζόμενα. Μπορείτε να παραλείψετε τον μη-γραμμικό μετασχηματισμό (Εξίσωση 3).

2.1 Αρχικά, χρησιμοποιείστε τη συνάρτηση `training.torch_train_val_split()` για να δημιουργήσετε validation set από το training set. Χρησιμοποιώντας την κλάση `early_stopper.EarlyStopper` διακόψτε την εκπαίδευση όταν το validation loss αυξάνεται διαρκώς για 5 εποχές.
2.2 Χρησιμοποιήστε την τελευταία έξοδο του LSTM h_N ως την αναπαράσταση του κειμένου u. Βασιστείτε στην υλοποίηση `models.LSTM` που παρέχεται.
Προσοχή: πρέπει να χρησιμοποιήσετε το πραγματικό τελευταίο timestep, εξαιρώντας τα zero-padded timesteps. Χρησιμοποιήστε τα πραγματικά μήκη των προτάσεων, όπως υπολογίσατε τον μέσο όρο στην άσκηση της προπαρασκευής.
2.3 Αξιοποιώντας την παράμετρο `bidirectional` στον κατασκευαστή της κλάσης, τρέξτε ξανά τα πειράματά σας χρησιμοποιώντας αμφίδρομο LSTM. Αναφέρετε την επίδοη του μοντέλου στο test set (accuracy, recall, f1-score).

Ερώτημα 3
Σε αυτό το ερώτημα θα χρησιμοποιήσετε ένα μηχανισμό attention για την κατηγοριοποίηση συναισθήματος.
3.1 Αξιοποιήστε το δοθέν μοντέλο `attention.SimpleSelfAttentionModel`, συμπληρώστε τα κενά και μετρήστε την επίδοσή του στο test set. Εφαρμόστε average-pooling στις αναπαραστάσεις των λέξεων πριν το τελευταίο layer, για να εξάγεται την αναπαράσταση της πρότασης. Ποια είναι η επίδοση του μοντέλου;
3.2 Τι είναι τα queries, keys κα values που υπάρχουν στη κλάση `attention.Head` και τα `position_embeddings` που ορίζονται στην `attention.SimpleSelfAttentionModel`; Μπορείτε να συμβουλευτείτε το tutorial "Let's build GPT: from scratch, in code, spelled out" για την απάντησή σας.

Ερώτημα 4
Σε αυτό το ερώτημα θα χρησιμοποιήσετε ένα μηχανισμό attention πολλαπλών κεφαλών (MultiHead-attention) για την κατηγοριοποίηση συναισθήματος.
Συμπληρώστε τα κενά στο μοντέλο `attention.MultiHeadAttentionModel`. Βασιστείτε στο `SimpleSelfAttentionModel` αλλά αξιοποιήστε το `MultiHeadAttention` για τον μηχανισμό προσοχής. Ποια είναι η επίδοση του μοντέλου;

Ερώτημα 5
Σε αυτό το ερώτημα θα χρησιμοποιήσετε έναν Transformer-Encoder για την κατηγοριοποίηση

[6] https://github.com/slp-ntua/slp-labs/tree/master/lab3

συναισθήματος. Διατηρείστε το average-pooling για την εξαγωγή της αναπαράστασης κάθε πρότασης.
Συμπληρώστε τα κενά στο μοντέλο `attention.TransformerEncoderModel`. Ποια είναι η διαφορά του σε σχέση με το `MultiHeadAttentionModel`; Πειραματιστείτε με διαφορετικές τιμές των παραμέτρων. Ποιες είναι οι default τιμές τους στην κλασική αρχιτεκτονική του Transformer; Τι επίδοση έχει το μοντέλο στο test set;

Ερώτημα 6
Σε αυτό το ερώτημα θα χρησιμοποιήσετε Pre-Trained Transformer μοντέλα για την κατηγοριοποίηση συναισθήματος. Συμπληρώστε το αρχείο `transfer_pretrained.py` και εκτελέστε για να αξιολογήσετε τα pre-trained μοντέλα. Επιλέξτε τουλάχιστον 3 μοντέλα, από εδώ[7], για κάθε dataset και επεκτείνετε κατάλληλα τον κώδικα. Συγκρίνετε τις επιδόσεις τους και αναφέρετε σε πίνακες τα αποτελέσματα.

Ερώτημα 7
Σε αυτό το ερώτημα θα εκπαιδεύσετε/κουρδίσετε (fine-tune) Pre-Trained Transformer μοντέλα για την κατηγοριοποίηση συναισθήματος. Αξιοποιήστε τον κώδικα που θα βρείτε στο αρχείο `finetune_pretrained.py` και εκτελέστε τοπικά με περιορισμένο dataset για λίγες εποχές (όπως δίνεται). Μεταφέρετε τον κώδικα σε notebook στο Google Colab[8], μετατρέψτε τον κατάλληλα και προχωρήστε σε fine-tuning με σκοπό τη βέλτιστη επίδοση. Δοκιμάστε τουλάχιστον 3 μοντέλα για κάθε dataset και αναφέρετε τα αποτελέσματα σε πίνακες.

Ερώτημα 8 (bonus)
Ο κώδικας για το tutorial "Let's build GPT: from scratch, in code, spelled out", που προαναφέρθηκε, είναι διαθέσιμος εδώ[9]. Συνδεθείτε στο ChatGPT[10] και ζητήστε:

- να σας εξηγήσει τον κώδικα σε λεπτομέρεια
- να αξιολογήσει τον κώδικα
- να ξαναγράψει κάποια τμήματά του (refactoring)

Αναφέρετε παραδείγματα από το διάλογο που κάνατε και σχολιάστε την απόδοση του στα παραπάνω.

Παραδοτέα

Στα ερωτήματα 1-5 μπορείτε να χρησιμοποιήσετε ένα από τα δύο datasets τα οποία αναφέρονται στην προπαρασκευή. Διαλέξτε όποιο θέλετε, απλά αναφέρετέ το στην αναφορά σας. Για κάθε μοντέλο, αναφέρετε στην αναφορά σας τις επιδόσεις του στις μετρικές: accuracy, F1-score (macro average), recall (macro average).

Σε ότι αφορά την βαθμολόγηση της άσκησης, δεν θα αξιολογηθείτε ως προς τις επιδόσεις του μοντέλου σας, αλλά ως προς την ορθότητα των απαντήσεων σας. Είστε ελεύθεροι να διαλέξετε ότι τιμές θέλετε για τις υπερπαραμέτρους του μοντέλου. Θα πρέπει να παραδώσετε τα εξής:

1. Σύντομη αναφορά (σε pdf) που θα περιέχει τις απαντήσεις σε κάθε ερώτημα και τα αντίστοιχα αποτελέσματα.
2. Κώδικας Python, συνοδευόμενος από σύντομα σχόλια.

Συγκεντρώστε τα (1) και (2) σε ένα .zip αρχείο και υποβάλετέ το εντός της προκαθορισμένης ημερομηνίας στο helios.

[7] https://huggingface.co/models
[8] https://colab.research.google.com/
[9] https://colab.research.google.com/drive/1JMLa53HDuA-i7ZBmqV7ZnA3c_fvtXnx-?usp=sharing
[10] https://chat.openai.com/

References

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. 02127.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Elman, 1990] Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 14(2):179–211. 08621.

[Hochreiter et al., 2001] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558. 18706.

[Jordan, 1997] Jordan, M. I. (1997). Serial Order: A Parallel Distributed Processing Approach. In *Advances in Psychology*, volume 121, pages 471–495. Elsevier. 01033.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.