Επεξεργασία Φωνής και Φυσικής Γλώσσας

3η εργαστηριακή άσκηση

Ηλίας Κουμουκέλης, el18157
Νικόλαος Παγώνας, el18175

Εισαγωγή

Σκοπός αυτής της εργαστηριακής άσκησης είναι να εμπλουτίσουμε την απλή αρχιτεκτονική της προπαρασκευής με RNNs, Transfer Learning και attention mechanisms. Και πάλι χρησιμοποιούμε τα `glove.6B.50d` embeddings, αλλά εδώ χρησιμοποιούμε μόνο το dataset `MR`.

Ερώτημα 1

1.1

Πλέον υπολογίζουμε την αναπαράσταση κάθε πρότασης u ως την συνένωση του μέσου όρου και του μεγίστου ανά διάσταση των embeddings της κάθε πρότασης $E = (e_1, e_2, \ldots, e_N)$.

$$u = [mean(E)||max(E)]$$

1.2

Με την συνεισφορά του μεγίστου (και όχι μόνο του μέσου όρου) μπορεί να δοθεί περισσότερη έμφαση σε ορισμένες λέξεις-κλειδιά, οι οποίες παίζουν σημαντικότερο ρόλο στην κατηγοριοποίηση με βάση το συναίσθημα. Αυτό είναι λογικό, αφού σε μία πρόταση οι περισσότερες λέξεις δεν αρκούν για να περιγράψουν αν το συναίσθημα της πρότασης είναι θετικό ή αρνητικό. Για παράδειγμα, οι προτάσεις:

- That was the **best** movie I have ever seen.
- That was the **worst** movie I have ever seen.

Διαφέρουν μόνο σε μία λέξη, ενώ οι υπόλοιπες 8 είναι ίδιες. Παρολαυτά, αυτή η μία λέξη διαφοροποιεί τις προτάσεις τόσο πολύ, που ουσιαστικά οι προτάσεις είναι άκρως αντίθετες σε συναίσθημα (πολύ θετικό έναντι πολύ αρνητικού). Μία ανάλυση που βασίζεται μόνο στον μέσο όρο θα μπορούσε να μειώσει την επίδραση που έχουν οι πολύ "φορτισμένες" λέξεις `best` και `worst`, ειδικά αν οι προτάσεις περιέχουν πολλές λέξεις. Με το max pooling όμως, οι σημαντικές λέξεις ξεχωρίζουν όπως πρέπει, ενώ ο συνδυασμός mean και max pooling αποτελεί μία μέση λύση, όπου οι λέξεις-κλειδιά ξεχωρίζουν μεν, αλλά δεν αναιρούν το νόημα όλης της υπόλοιπης πρότασης.

Ερώτημα 2

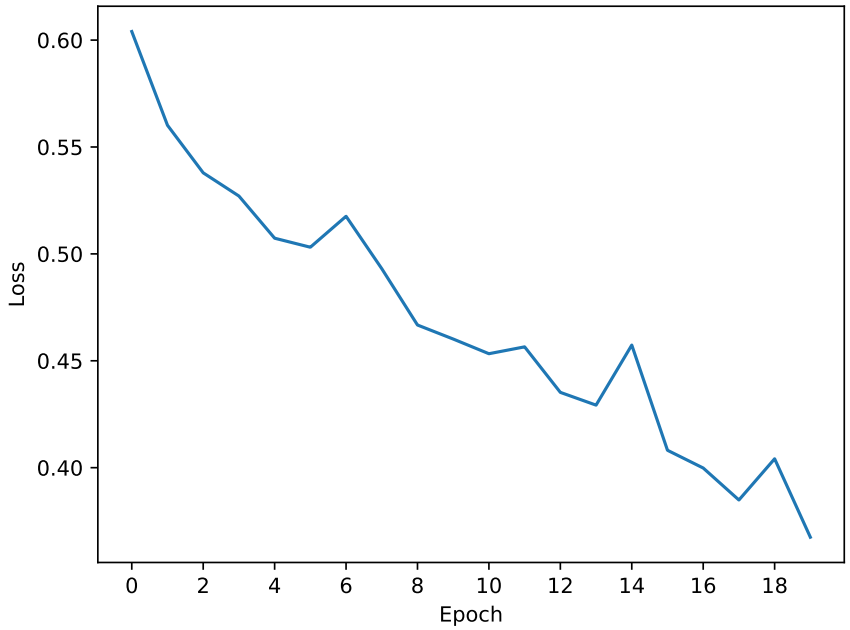
Σε αυτό το ερώτημα χρησιμοποιούμε ένα LSTM για την κωδικοποίηση της πρότασης, το οποίο θα διαβάζει τα word embeddings e_i και θα παράγει μία νέα αναπαράσταση για κάθε λέξη h_i, η οποία θα λαμβάνει υπόψιν της και τα συμφραζόμενα.

2.1

Εδώ χρησιμοποιούμε την τελευταία έξοδο του LSTM h_N ως την αναπαράσταση του κειμένου u. Να σημειωθεί ότι χρησιμοποιούμε ως τελευταίο timestep το πραγματικό (εξαιρούμε δηλαδή το zero padding, όπως στην προπαρασκευή). Η υλοποίηση βρίσκεται στην κλάση `Model21`, στο `models.py`.

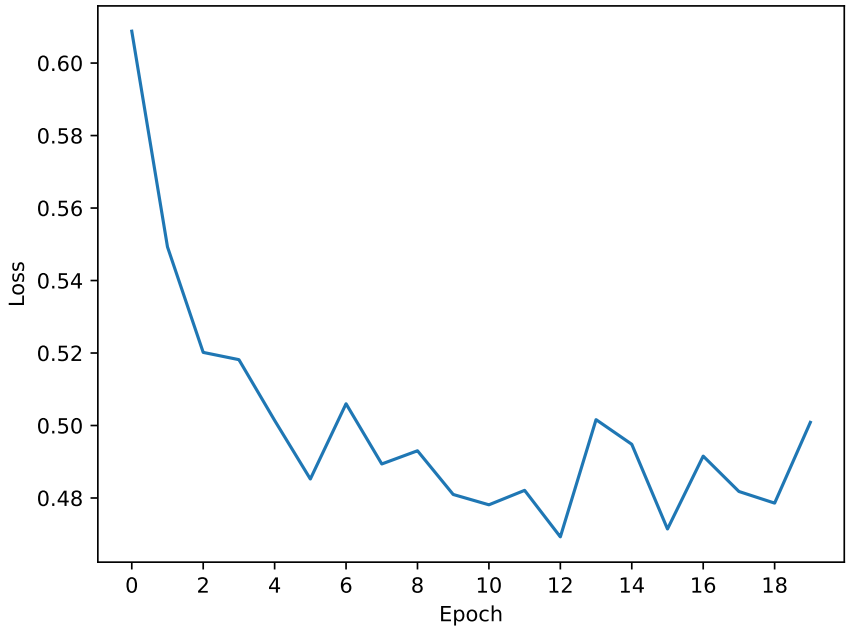
Τα αποτελέσματα είναι τα εξής:

- Training dataset:
  - Accuracy: 0.828
  - Recall: 0.828
  - F1 Score: 0.827
  - Loss:



- Test dataset:
  - Accuracy: 0.769
  - Recall: 0.454
  - F1 Score: 0.496

- Loss:



2.2

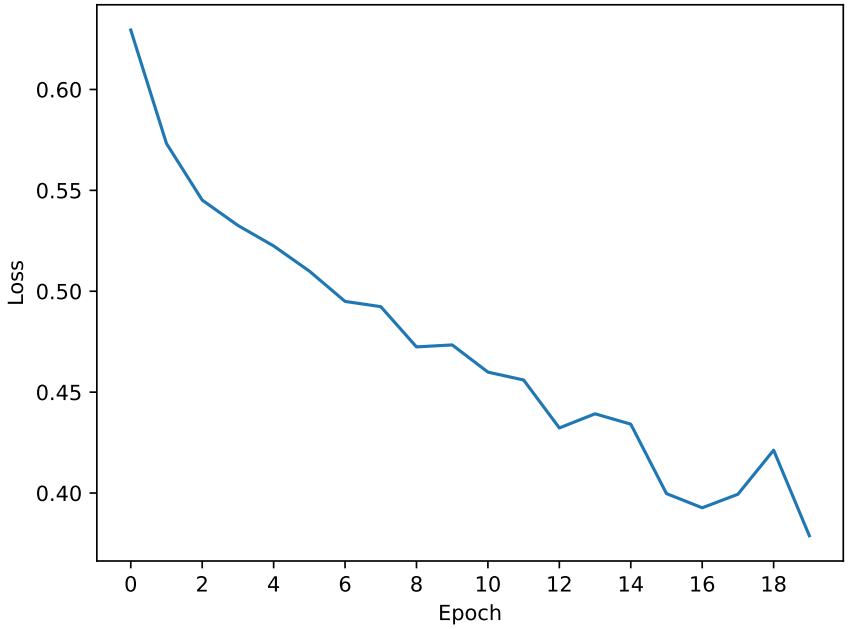
Εδώ χρησιμοποιούμε ως αναπαράσταση του κειμένου u την συνένωση:

$$u = [h_N||mean(E)||max(E)]$$

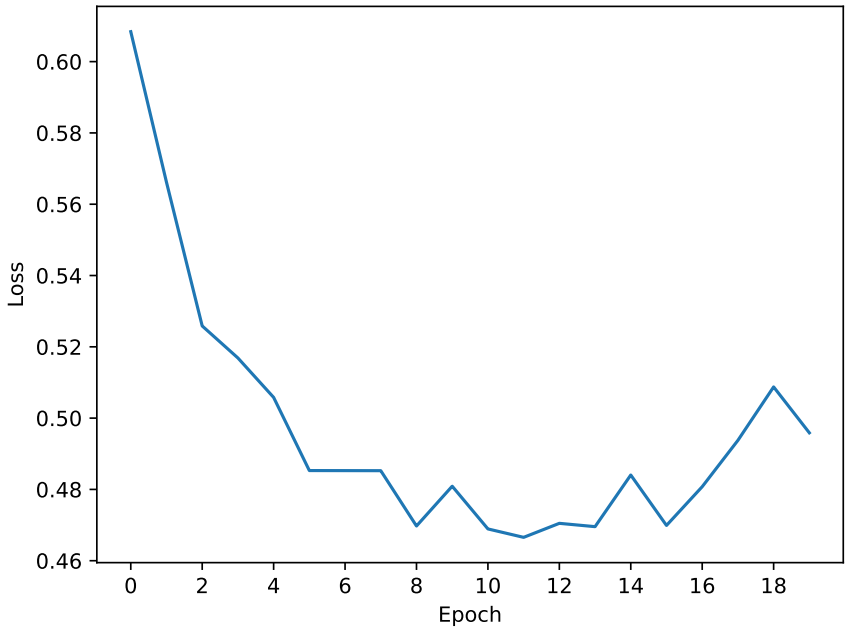
Η υλοποίηση βρίσκεται στην κλάση `Model22`, στο `models.py`.

Τα αποτελέσματα είναι τα εξής:

- Training dataset:
  - Accuracy: 0.840
  - Recall: 0.839
  - F1 score: 0.839
  - Loss:



- Test dataset:
  - Accuracy: 0.778
  - Recall: 0.482
  - F1 score: 0.503
  - Loss:



Ερώτημα 3

Σε αυτό το ερώτημα χρησιμοποιούμε ένα μηχανισμό attention. Παίρνουμε έτοιμη την υλοποίηση που βρίσκεται σε αυτό το gist.

3.1

Χρησιμοποιούμε τον μηχανισμό attention, για να υπολογίσουμε την αναπαράσταση ενός κειμένου, ως το σταθμισμένο άθροισμα των word embeddings:

$$v_i = tanh(We_i + b)$$

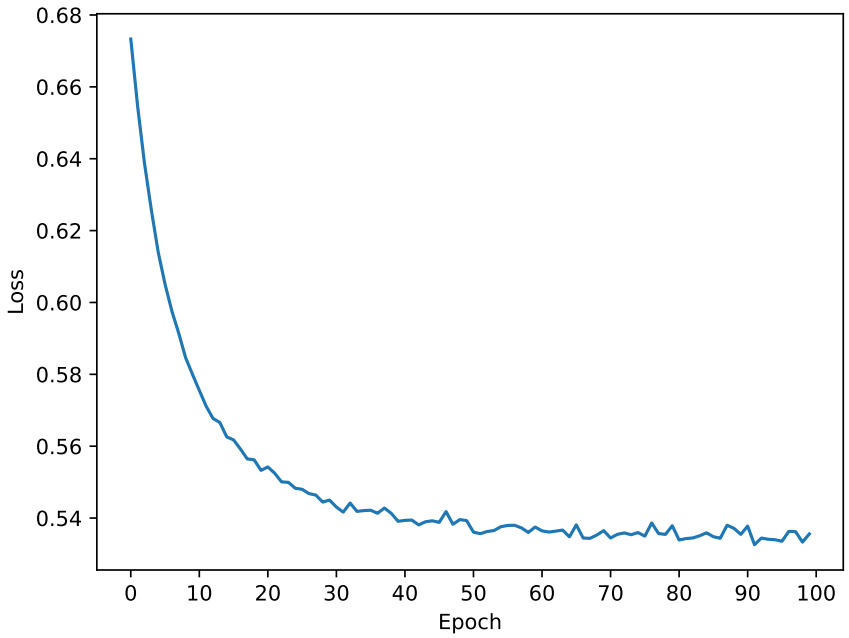
$$a_i = \frac{exp(v_i)}{\sum_{t=1}^{N} exp(u_t)}$$

$$u = \sum_{i=1}^{N} a_i e_i$$

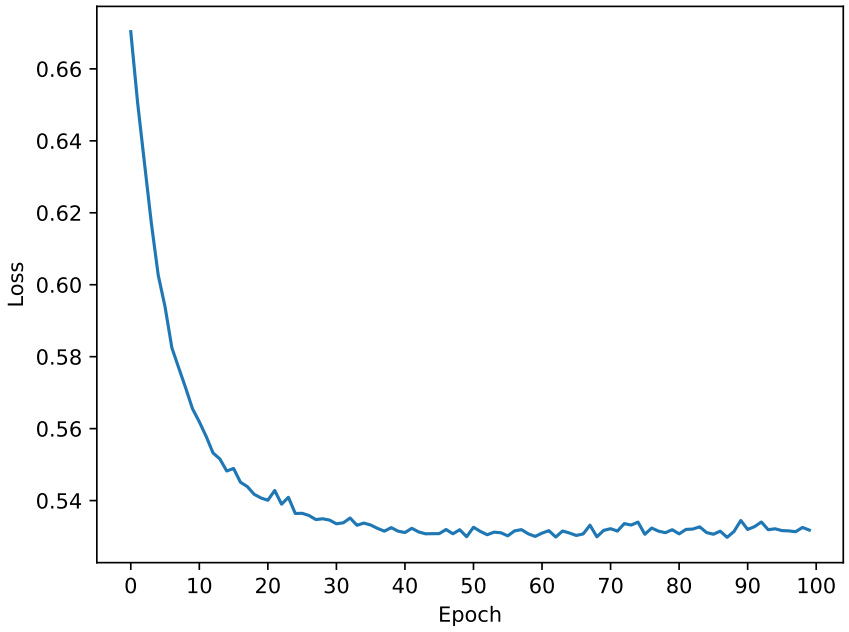
Η υλοποίηση βρίσκεται στην κλάση `Model31`, στο `models.py`.

Τα αποτελέσματα είναι τα εξής:

- Training dataset:
  - Accuracy: 0.726
  - Recall: 0.726
  - F1 Score: 0.724
  - Loss:



- Test dataset:
  - Accuracy: 0.734
  - Recall: 0.432
  - F1 Score: 0.478
  - Loss:



3.2

Πάλι χρησιμοποιούμε τον μηχανισμό attention για να υπολογίσουμε την αναπαράσταση ενός κειμένου, αλλά αυτή τη φορά ως το σταθμισμένο άθροισμα των εξόδων ενός LSTM.

$$v_i = tanh(Wh_i + b)$$

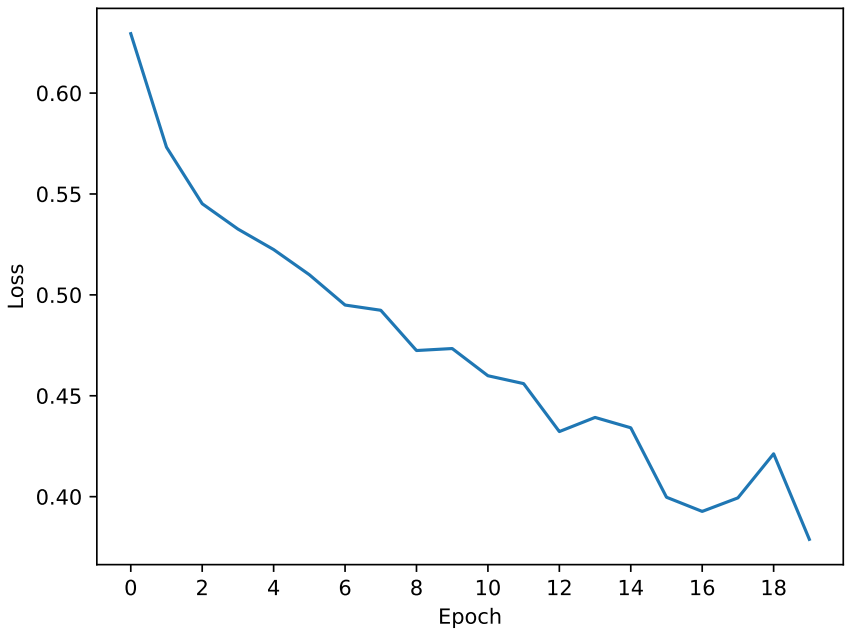
$$a_i = \frac{exp(v_i)}{\sum_{t=1}^{N} exp(u_t)}$$

$$u = \sum_{i=1}^{N} a_i h_i$$

Η υλοποίηση βρίσκεται στην κλάση `Model32`, στο `models.py`.

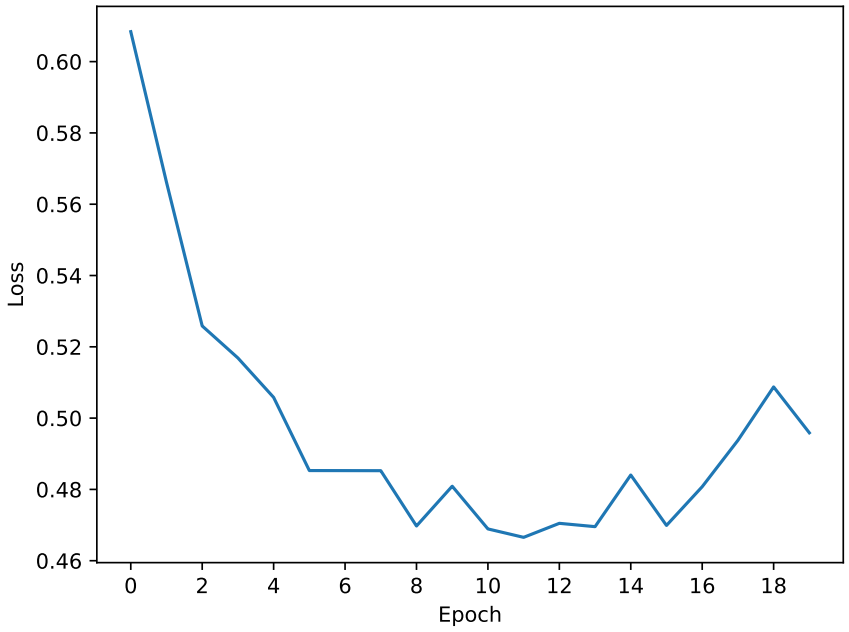
Τα αποτελέσματα είναι τα εξής:

- Training dataset:
  - Accuracy: 0.828
  - Recall: 0.828
  - F1 Score: 0.826
  - Loss:



- Test dataset:

- Accuracy: 0.745
- Recall: 0.438
- F1 Score: 0.485
- Loss:



Ερώτημα 4

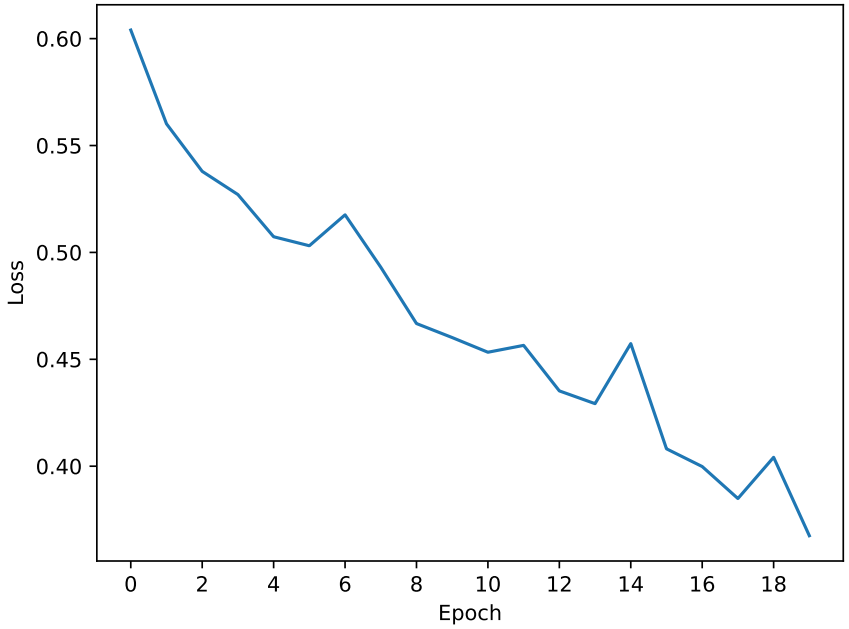
Σε αυτό το ερώτημα υλοποιούμε τα ζητούμενα του ερωτήματος 3, αλλά αυτή τη φορά χρησιμοποιούμε ένα **αμφίδρομο** LSTM.

4.1

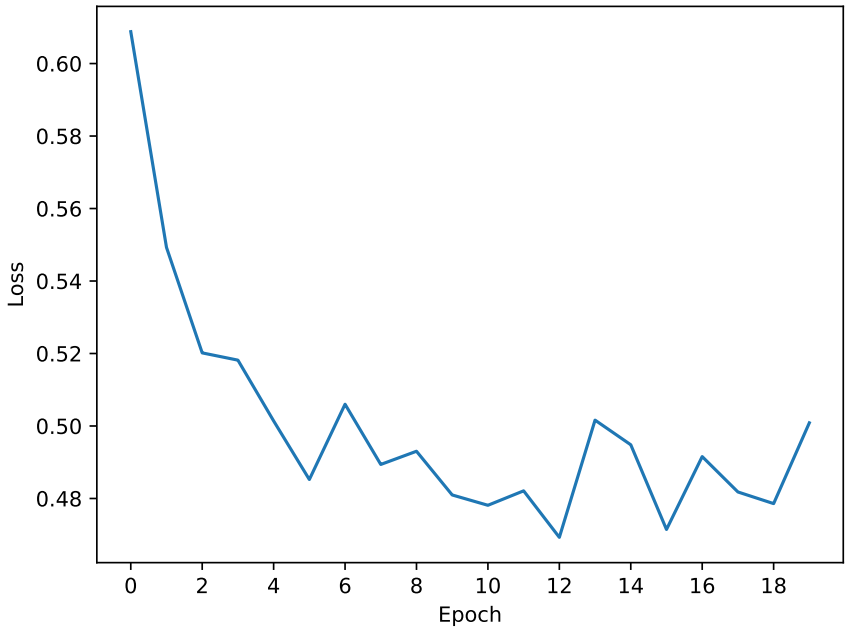
Αυτό το ερώτημα είναι όμοιο με το 2.2, αλλά με αμφίδρομο LSTM. Η υλοποίηση βρίσκεται στην κλάση `Model41`, στο `models.py`.

Τα αποτελέσματα είναι τα εξής:

- Training dataset:
  - Accuracy: 0.831
  - Recall: 0.832
  - F1 score: 0.828
  - Loss:



- Test dataset:
  - Accuracy: 0.782
  - Recall: 0.523
  - F1 score: 0.521
  - Loss:

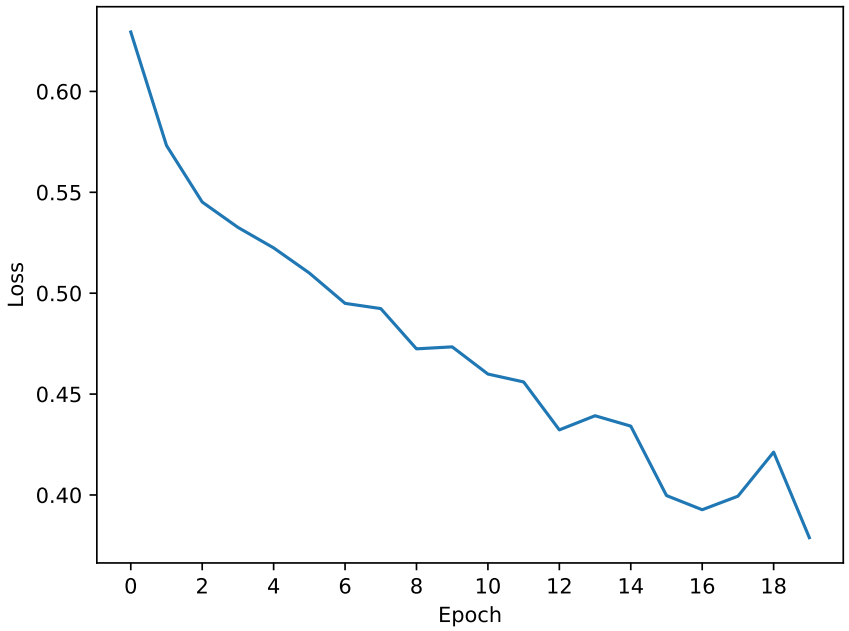


4.2

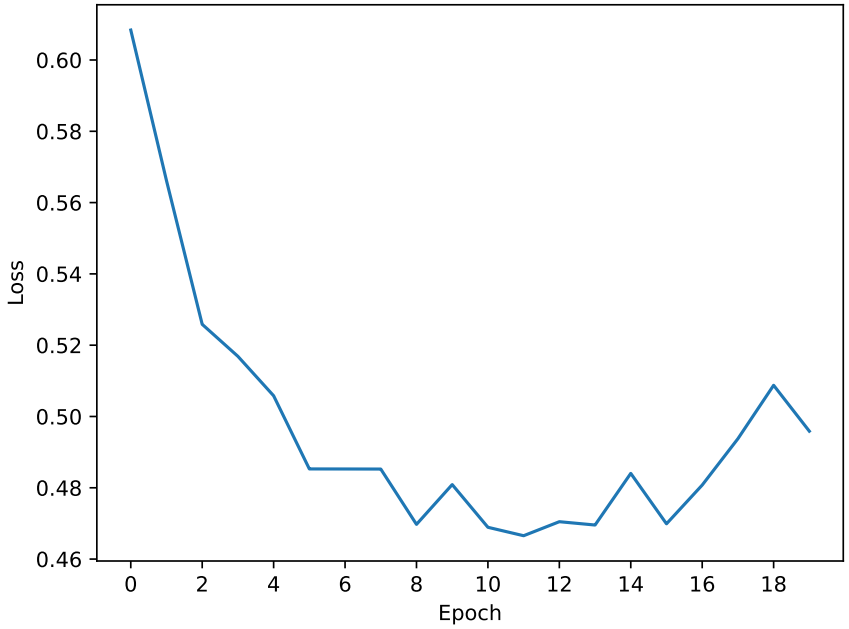
Αυτό το ερώτημα είναι όμοιο με το 3.2, αλλά με αμφίδρομο LSTM. Η υλοποίηση βρίσκεται στην κλάση `Model42`, στο `models.py`.

Τα αποτελέσματα είναι τα εξής:

- Training dataset:
  - Accuracy: 0.821
  - Recall: 0.819
  - F1 score: 0.819
  - Loss:



- Test dataset:
  - Accuracy: 0.790
  - Recall: 0.781
  - F1 score: 0.783
  - Loss:



Ερώτημα 5

5.1

...

5.2

...

5.3

...

Ερώτημα 6

6.1

...

6.2

Τα BoW χαρακτηριστικά θα μπορούσουν να οδηγήσουν σε καλύτερα αποτελέσματα σε σχέση με αναπαραστάσεις όπως ο μέσος όρος των word embeddings, σε περιπτώσεις όπου:

- Θέλουμε να δημιουργήσουμε baseline models, αφού πολύ εύκολα και γρήγορα, με έτοιμες συναρτήσεις βιβλιοθηκών, και χωρίς περίπλοκες αναπαραστάσεις, μπορούμε να φτιάξουμε μία απλοϊκή μορφή μοντέλων για κατηγοριοποίηση κειμένων.

- Το dataset είναι μικρό σε μέγεθος και έχουμε να κάνουμε με domain-specific context. Όταν τα κείμενα περιέχουν εξειδικευμένους/τεχνικούς όρους, τα pretrained embeddings που υπάρχουν δεν αποδίδουν τόσο καλά, γιατί είναι πολύ γενικά, και ταυτόχρονα είναι πιο δύσκολο να βρούμε τόσο εξειδικευμένα pretrained embeddings. Έτσι, προτιμάται η αναπαράσταση με χαρακτηριστικά tfidf.