Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ροή Σ: Επεξεργασία Φωνής και Φυσικής Γλώσσας

($10^{ο}$ Εξάμηνο)

Εργαστηριακή Άσκηση 3

Δημήτρης Μπακάλης 03118163

Εισαγωγή:

Στη συγκεκριμένη εργαστηριακή άσκηση θα ασχοληθούμε με προβλήματα επεξεργασίας φυσικής γλώσσας. Πιο συγκεκριμένα, χρησιμοποιώντας τη βιβλιοθήκη PyTorch, θα χρησιμοποιήσουμε διάφορα μοντέλα (DNNs, LSTMS, Transformer-Encoder models) για την κατηγοριοποίηση κειμένων με βάση το συναίσθημα (sentiment analysis). Οι αναπαραστάσεις των λέξεων θα προκύπτουν από προεκπαιδευμένα embeddings (διαστάσεων 50 και 300), ενώ τα datasets που θα αξιοποιηθούν θα είναι τα ακόλουθα:

- *Sentence Polarity Dataset (MR):* Το dataset αυτό περιέχει 5331 θετικές και 5331 αρνητικές κριτικές ταινιών, από το Rotten Tomatoes και είναι binary-classification πρόβλημα (positive, negative).

- *Semeval 2017 Task4-A3:* Το dataset αυτό περιέχει tweets τα οποία είναι κατηγοριοποιημένα σε 3 κλάσεις (positive, negative, neutral) με 49570 παραδείγματα εκπαίδευσης και 12284 παραδείγματα αξιολόγησης.

Βήματα προπαρασκευής

Μέρος 1: Προεπεξεργασία δεδομένων

Αρχικά, θα επεξεργαστούμε τα labels του dataset, έτσι ώστε κάθε label να αντιστοιχεί σε έναν συγκεκριμένο αριθμό. Αυτό θα γίνει μέσω του LabelEncoder της scikit-learn και τα δεδομένα θα μετατραπούν ως εξής:

- *MR dataset:*

```
the rock is destined to be the 21st century's new " conan " and that he's going to make a splash even greater than arnold schwarzenegger , jean-claud van damme or steven segal . 1
the gorgeously elaborate continuation of " the lord of the rings " trilogy is so huge that a column of words cannot adequately describe co-writer/director peter jackson's expanded vision of j . r . r . tolkien's middle-earth . 1
effective but too-tepid biopic 1
if you sometimes like to go to the movies to have fun , wasabi is a good place to start . 1
emerges as something rare , an issue movie that's so honest and keenly observed that it doesn't feel like one . 1
the film provides some great insight into the neurotic mindset of all comics -- even those who have reached the absolute top of the game . 1
offers that rare combination of entertainment and education . 1
perhaps no picture ever made has more literally showed that the road to hell is paved with good intentions . 1
steers turns in a snappy screenplay that curls at the edges ; it's so clever you want to hate it . but he somehow pulls it off . 1
take care of my cat offers a refreshingly different slice of asian cinema . 1
```

- *Semeval 2017 dataset:*

```
"Picturehouse's, Pink Floyd's, 'Roger Waters: The Walll – opening 29 Sept is now making waves. Watch the trailer on Rolling Stone – look..." 1
Order Go Set a Watchman in store or through our website before Tuesday and get it half price! #GSAW @GSAWatchmanBook https://t.co/KET6EGD1an 1
"If these runway renovations at the airport prevent me from seeing Taylor Swift on Monday, Bad Blood will have a new meaning." 0
"If you could ask an onstage interview question at Miss USA tomorrow, what would it be?" 1
A portion of book sales from our Harper Lee/Go Set a Watchman release party on Mon. 7/13 will support @CAP_Tulsa and the great work they do. 2
Excited to read "Go Set a Watchman" on Tuesday. But can it possibly live up to "To Kill a Mockingbird?" Any opinions? 2
"Watching Miss USA tomorrow JUST to see @TravisGarland perform, I'm obsessed with his voice" 2
"Tune-in for the 2015 MISS USA Pageant on ReelzChannel on Sunday, July 12 at 8p ET/5p PT. Contestants from all 50... http://t.co/M3kJowOvQ1" 1
Call for reservations for lunch or dinner tomorrow (yep Sunday!). Happy to accommodate guests in town for the MISS USA Pageant 346-5100 2
"Miss Universe Org prez tells me #Trump won't attend Sunday's Miss USA event He's missed some in the past, but he said recently he'd be here" 1
```

Στη συνέχεια, μέσω της βιβλιοθήκης NTLK (word_tokenization), θα πραγματοποιήσουμε τη λεκτική ανάλυση (tokenization) των προτάσεων. Επομένως, οι προτάσεις στα δεδομένα εκπαίδευσης θα μετατραπούν ως εξής:

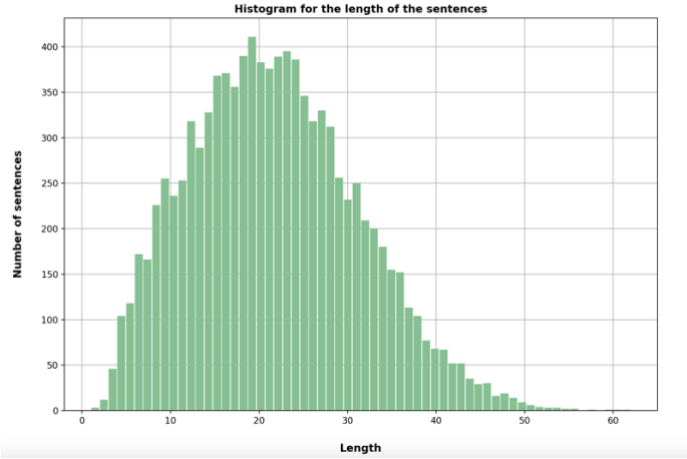
- *MR dataset:*

```
['the', 'rock', 'is', 'destined', 'to', 'be', 'the', '21st', 'century', "'s", 'new', '``', 'conan', "''", 'and', 'that', 'he', "'s", 'going', 'to', 'make', 'a', 'splash', 'even', 'greater', 'than', 'arnold', 'schwarzenegger', ',', 'jean-claud', 'van', 'damme', 'or', 'steven', 'segal', '.']
['the', 'gorgeously', 'elaborate', 'continuation', 'of', '``', 'the', 'lord', 'of', 'the', 'rings', "''", 'trilogy', 'is', 'so', 'huge', 'that', 'a', 'column', 'of', 'words', 'can', 'not', 'adequately', 'describe', 'co-writer/director', 'peter', 'jackson', "'s", 'expanded', 'vision', 'of', 'j', '.', 'r', '.', 'r', '.', 'tolkien', "'s", 'middle-earth', '.']
['effective', 'but', 'too-tepid', 'biopic']
['if', 'you', 'sometimes', 'like', 'to', 'go', 'to', 'the', 'movies', 'to', 'have', 'fun', ',', 'wasabi', 'is', 'a', 'good', 'place', 'to', 'start', '.']
['emerges', 'as', 'something', 'rare', ',', 'an', 'issue', 'movie', 'that', "'s", 'so', 'honest', 'and', 'keenly', 'observed', 'that', 'it', 'does', "n't", 'feel', 'like', 'one', '.']
['the', 'film', 'provides', 'some', 'great', 'insight', 'into', 'the', 'neurotic', 'mindset', 'of', 'all', 'comics', '--', 'even', 'those', 'who', 'have', 'reached', 'the', 'absolute', 'top', 'of', 'the', 'game', '.']
['offers', 'that', 'rare', 'combination', 'of', 'entertainment', 'and', 'education', '.']
['perhaps', 'no', 'picture', 'ever', 'made', 'has', 'more', 'literally', 'showed', 'that', 'the', 'road', 'to', 'hell', 'is', 'paved', 'with', 'good', 'intentions', '.']
['steers', 'turns', 'in', 'a', 'snappy', 'screenplay', 'that', 'curls', 'at', 'the', 'edges', ';', 'it', "'s", 'so', 'clever', 'you', 'want', 'to', 'hate', 'it', '.', 'but', 'he', 'somehow', 'pulls', 'it', 'off', '.']
['take', 'care', 'of', 'my', 'cat', 'offers', 'a', 'refreshingly', 'different', 'slice', 'of', 'asian', 'cinema', '.']
```

- *Semeval 2017 dataset:*

```
['``', 'picturehouse', "'s", ',', 'pink', 'floyd', "'s", ',', "'roger", 'waters', ':', 'the', 'walll', '–', 'opening', '29', 'sept', 'is', 'now', 'making', 'waves', '.', 'watch', 'the', 'trailer', 'on', 'rolling', 'stone', '–', 'look', '...', "''"]
['order', 'go', 'set', 'a', 'watchman', 'in', 'store', 'or', 'through', 'our', 'website', 'before', 'tuesday', 'and', 'get', 'it', 'half', 'price', '!', '#', 'gsaw', '@', 'gsawatchmanbook', 'https', ':', '//t.co/ket6egd1an']
['``', 'if', 'these', 'runway', 'renovations', 'at', 'the', 'airport', 'prevent', 'me', 'from', 'seeing', 'taylor', 'swift', 'on', 'monday', ',', 'bad', 'blood', 'will', 'have', 'a', 'new', 'meaning', '.', "''"]
['``', 'if', 'you', 'could', 'ask', 'an', 'onstage', 'interview', 'question', 'at', 'miss', 'usa', 'tomorrow', ',', 'what', 'would', 'it', 'be', '?', "''"]
['a', 'portion', 'of', 'book', 'sales', 'from', 'our', 'harper', 'lee/go', 'set', 'a', 'watchman', 'release', 'party', 'on', 'mon', '.', '7/13', 'will', 'support', '@', 'cap_tulsa', 'and', 'the', 'great', 'work', 'they', 'do', '.']
['excited', 'to', 'read', '``', 'go', 'set', 'a', 'watchman', "''", 'on', 'tuesday', '.', 'but', 'can', 'it', 'possibly', 'live', 'up', 'to', '``', 'to', 'kill', 'a', 'mockingbird', '?', "''", 'any', 'opinions', '?']
['``', 'watching', 'miss', 'usa', 'tomorrow', 'just', 'to', 'see', '@', 'travisgarland', 'perform', ',', 'i', "'m", 'obsessed', 'with', 'his', 'voice', "''"]
['``', 'tune-in', 'for', 'the', '2015', 'miss', 'usa', 'pageant', 'on', 'reelzchannel', 'on', 'sunday', ',', 'july', '12', 'at', '8p', 'et/5p', 'pt', '.', 'contestants', 'from', 'all', '50', '...', 'http', ':', '//t.co/m3kjowovq1', "''"]
['call', 'for', 'reservations', 'for', 'lunch', 'or', 'dinner', 'tomorrow', '(', 'yep', 'sunday', '!', ')', '.', 'happy', 'to', 'accommodate', 'guests', 'in', 'town', 'for', 'the', 'miss', 'usa', 'pageant', '346-5100']
['``', 'miss', 'universe', 'org', 'prez', 'tells', 'me', '#', 'trump', 'wo', "n't", 'attend', 'sunday', "'s", 'miss', 'usa', 'event', 'he', "'s", 'missed', 'some', 'in', 'the', 'past', ',', 'but', 'he', 'said', 'recently', 'he', "'d", 'be', 'here', "''"]
```

Τέλος, θα κωδικοποιήσουμε τις λέξεις με βάση κάποια προεκπαιδευμένα embeddings. Στη συγκεκριμένη περίπτωση θα χρησιμοποιηθούν τα GloVe embeddings διαστάσεων 50 και 300. Σε περίπτωση που δεν υπάρχει η αντιστοίχιση για κάποιο token των δεδομένων, τότε αυτό αντιστοιχίζεται με το token <unk> (unknown), το οποίο είναι ένα διάνυσμα με μηδενικά. Επιπλέον, εφόσον στο πλαίσιο της προπαρασκευής θα εξετάσουμε μόνο ένα απλό νευρωνικό δίκτυο (baseline DNN), θέλουμε να τα δεδομένα εισόδου να έχουν ίδιο μήκος, ώστε να μπορούν να εκτελεστούν οι πολλαπλασιασμοί πινάκων. Για αυτό τον σκοπό θα επιλέξουμε έναν αριθμό, ο οποίος θα ορίζει το μέγιστο μήκος μιας πρότασης, οπότε όσες προτάσεις έχουν μικρότερο μήκος θα γίνονται zero padded, ενώ όσες έχουν μεγαλύτερο μήκος, θα αγνοούνται τα τελευταία tokens. Για να καθορίσουμε την τιμή του μεγίστου μήκους των προτάσεων, αρχικά, θα προβάλλουμε το ιστόγραμμα των προτάσεων με τα μήκη τους, ώστε να έχουμε πιο ξεκάθαρη εικόνα για τους outliers.



Επομένως, βάσει του παραπάνω ιστογράμματος θα επιλέξουμε μέγιστο μήκος ίσο με 40 για τα πειράματα της συγκεκριμένης εργαστηριακής άσκησης.

Ωστόσο, σε κάθε περίπτωση είναι σημαντικό να γνωρίζουμε και το κανονικό μήκος κάθε πρότασης. Για αυτόν τον λόγο, η μέθοδος __getitem__ της κλάσης SentenceDataset θα επιστρέφει:

1. Την κωδικοποιημένη μορφή της πρότασης (με βάση το index του vocabulary).
2. Το κωδικοποιημένο label της πρότασης.
3. Το πραγματικό μήκος της πρότασης.

Παρακάτω παρατίθενται μερικά παραδείγματα από το τι επιστρέφει η __getitem__:

```
(tensor([    29,  98405,     10,      2,   5492,   8737,     10,      2, 400001,
          2879,     46,      1, 400001,     12,    876,   1264,  15776,     15,
           115,    434,   4978,      3,   1717,      1,   8719,     14,   4538,
          2156,     12,    663,    435,     28,      0,      0,      0,      0,
             0,      0,      0,      0]), 1, 32)
(tensor([   461,    243,    209,      8,  41915,      7,   1613,     47,    132,
           163,   2558,    107,    172,      6,    170,     21,    344,    627,
           806,   2750, 400001,  17528, 400001, 123043,     46, 400001,      0,
             0,      0,      0,      0,      0,      0,      0,      0,      0,
             0,      0,      0,      0]), 1, 26)
(tensor([   29,    84,   159,  6422, 14873,    23,     1,   923,  1450,   286,
          26,  2661,  2486,  6597,    14,   184,     2,   979,  1455,    44,
          34,     8,    51,  2302,     3,    28,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,     0]), 0, 26)
(tensor([   29,    84,    82,    95,  1713,    30, 15486,   963,   996,    23,
        2215,  2397,  4003,     2,   103,    55,    21,    31,   189,    28,
           0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,     0]), 1, 20)
(tensor([    8,  2940,     4,   540,   527,     26,   163,  6984, 400001,
         209,     8,  41915,   714,   166,     14,  14002,     3, 400001,
          44,   281,  17528, 400001,     6,      1,   354,   162,     40,
          89,     3,     0,     0,     0,      0,     0,     0,      0,
           0,     0,     0,     0]), 2, 29)
```

Μέρος 2: Μοντέλο

Στο συγκεκριμένο βήμα, θα δημιουργήσουμε το νευρωνικό δίκτυο, το οποίο θα κατηγοριοποιεί τις προτάσεις του dataset.

Αρχικά, θα δημιουργήσουμε ένα embedding layer, από το οποίο θα περνάνε οι προτάσεις, ώστε να κωδικοποιηθούν σε συνεχείς διανυσματικές αναπαραστάσεις . Τα βάρη αυτού του layer θα αρχικοποιηθούν βάσει των προεκπαιδευμένων embeddings που αναφέρθηκαν παραπάνω και δεν θα είναι εκπαιδεύσιμα.

Ερωτήσεις:

- Γιατί αρχικοποιούμε το embedding layer με τα προ-εκπαιδευμένα word embeddings?

Αρχικοποιούμε το embeddings layer με τα pretrained GloVe embeddings, διότι αυτά έχουν εκπαιδευτεί με πολύ μεγαλύτερο πλήθος δεδομένων και γενικεύουν πολύ καλύτερα από τα αντίστοιχα embeddings που θα προκύπταν σε περίπτωση εκπαίδευσής τους με τυχαία αρχικοποίηση.

- Γιατί κρατάμε παγωμένα τα βάρη του embedding layer κατά την εκπαίδευση?

Όπως αναφέρθηκε και παραπάνω, λόγω της μεγαλύτερο όγκου δεδομένων, στον οποίο έχουν εκπαιδευτεί τα GloVe embeddings, αυτά αποτελούν πολύ καλές αναπαραστάσεις των tokens και το trade-off μεταξύ του fine-tuning των embeddings και της ταχύτητας της εκπαίδευσης του μοντέλου δεν αξίζει. Επιπλέον, σε περίπτωση που επιλέγαμε να εκπαιδεύσουμε τα embeddings θα μπορούσαμε να είχαμε και overfitting.

Στη συνέχεια, θα δημιουργήσουμε τα output layers. Πιο συγκεκριμένα, αφού οι λέξεις κωδικοποιηθούν σε embeddings, θα περάσουν από ένα layer, το οποίο θα μετασχηματίσει τις αναπαραστάσεις σε κάποιες νέες 1000 διαστάσεων. Στις νέες αυτές αναπαραστάσεις θα εφαρμοστεί μία μη γραμμική ενεργοποίηση (ReLU στη συγκεκριμένη περίπτωση), προτού προβληθούν στις κλάσεις, μέσω του τελευταίου layer.

Ερωτήσεις:

- Γιατί βάζουμε μία μη γραμμική συνάρτηση ενεργοποίησης στο προτελευταίο layer; Τι διαφορά θα είχε αν είχαμε 2 ή περισσότερους γραμμικούς μετασχηματισμούς στη σειρά;

Η μη γραμμική συνάρτηση ενεργοποίησης χρησιμοποιείται επειδή εισάγει μία μη γραμμικότητα στο μοντέλο, η οποία του επιτρέπει να μπορεί να μάθει πιο πολύπλοκες σχέσεις για τα δεδομένα. Σε περίπτωση που υπήρχαν απλά 2 ή περισσότεροι γραμμικοί μετασχηματισμοί, η ενεργοποίηση θα ήταν ισοδύναμη με έναν γραμμικό μετασχηματισμό, με αποτέλεσμα η ικανότητα του μοντέλου να μαθαίνει σχέσεις μεταξύ των δεδομένων να περιοριστεί σημαντικά.

Τέλος, θα υλοποιήσουμε το forward pass του νευρωνικού δικτύου, με βάση τα layers που σχεδιάσαμε παραπάνω. Αναλυτικότερα, οι μετασχηματισμοί, σε συνδυασμό με τις διαστάσεις των δεδομένων μετά από κάθε μετασχηματισμό, φαίνονται παρακάτω:

1. Κωδικοποίηση σε embeddings, όπου η είσοδος θα είναι διαστάσεων (batch_size, max_length) και η έξοδος διαστάσεων (batch_size, max_length, embeddings_dim).

2. Δημιουργία μίας νέας αναπαράστασης κάθε πρότασης μέσω ενός average pooling των επιμέρους embeddings κάθε λέξης ανά διάσταση με είσοδο διαστάσεων (batch_size, max_length, embeddings_dim) και έξοδο διαστάσεων (batch_size, embeddings_dim). Αξίζει να σημειωθεί πως για την εξαγωγή του μέσου όρου κάθε διάστασης λαμβάνονται υπόψη μόνο οι τιμές των κανονικών λέξεων κι όχι των zero padded timesteps.

3. Δημιουργία μίας νέας αναπαράστασης μέσω ενός linear layer εισόδου (batch_size, embeddings_dim) και εξόδου (batch_size, 1000).

4. Δημιουργία μίας νέας αναπαράστασης ίδιων διαστάσεων, μέσω της εφαρμογής μίας μη γραμμικής συνάρτησης ενεργοποίησης (ReLU).

5. Δημιουργία της τελικής αναπαράστασης μέσω ενός linear layer εισόδου (batch_size, 1000) και εξόδου (batch_size, n_classes).

Ερωτήσεις:

- Αν θεωρήσουμε ότι κάθε διάσταση του embedding χώρου αντιστοιχεί σε μία αφηρημένη έννοια, μπορείτε να δώσετε μία διαισθητική ερμηνεία για το τι περιγράφει η αναπαράσταση που φτιάξατε (κέντρο-βάρους);

Η τελική αναπαράσταση κάθε πρότασης είναι ο μέσος όρος ανά διάσταση των αναπαραστάσεων των λέξεων της πρότασης. Επομένως, διαισθητικά, η αναπαράσταση κάθε πρότασης αποτελείται από τη μέση αφηρημένη έννοια που αντιστοιχεί σε κάθε διάσταση συμπεριλαμβάνοντας την πληροφορία όλων των επιμέρους λέξεων.

- Αναφέρετε πιθανές αδυναμίες της συγκεκριμένης προσέγγισης για να αναπαραστήσουμε κείμενα.

Τα βασικά μειονεκτήματα μιας τέτοιας προσέγγισης είναι πως η τελική αναπαράσταση χάνει την πληροφορία για τη σειρά τον λέξεων (Bag of Words), ενώ δίνεται, επίσης, ίδια έμφαση σε όλες τις λέξεις, κάτι το οποίο στην πραγματικότητα δεν ισχύει.

Μέρος 3: Διαδικασία Εκπαίδευσης

Σε αυτό το μέρος θα ασχοληθούμε με την εκπαίδευση του νευρωνικού δικτύου που δημιουργήσαμε.

Προκειμένου να προχωρήσουμε στην εκπαίδευση του δικτύου, αρχικά, θα ορίσουμε τους dataloaders των train και test δεδομένων ώστε να οργανώσουμε τα δεδομένα μας σε mini-batches, χρησιμοποιώντας την κλάση DataLoader του torch.

Ερωτήσεις:

- Τι συνέπειες έχουν τα μικρά και μεγάλα mini-batches στην εκπαίδευση των μοντέλων;

Τα μικρά mini-batches συνήθως προσθέτουν ένα είδος θορύβου στα δεδομένα, επειδή λαμβάνουμε ένα μικρό μέρος των δεδομένων, οπότε οι τιμές των gradients δεν είναι τόσο ακριβείς. Για τον ίδιο λόγο είναι υπολογιστικά «ακριβή» η χρήση μικρών mini-batches, καθώς χρειάζονται περισσότεροι υπολογισμοί gradients. Αντίθετα, μεγάλα mini-batches συνεπάγονται πιο γρήγορη εκπαίδευση, αλλά και μοντέλα, τα οποία είναι λιγότερο ανθεκτικά στο overfitting για τους ίδιους λόγους.

- Συνήθως ανακατεύουμε την σειρά των mini-batches στα δεδομένα εκπαίδευσης σε κάθε εποχή. Μπορείτε να εξηγήσετε γιατί;

Ο κύριος λόγος για τον οποίο ανακατεύεται η σειρά των mini-batches σε κάθε εποχή είναι για να σπάσουν οι εξαρτήσεις του μοντέλου σχετικά με τη σειρά των δεδομένων, με αποτέλεσμα να έχουμε πιο σταθερή σύγκλιση και να αποφύγουμε το overfitting.

Στη συνέχεια, θα προχωρήσουμε στις επιλογή του loss function και του optimizer, βάσει των οποίων θα ανανεώνονται τα βάρη του δικτύου. Πιο συγκεκριμένα, στη συγκεκριμένη περίπτωση, ως loss function θα χρησιμοποιηθεί το cross entropy για το dataset Semeval 2017 και binary cross entropy για το dataset MR. Επιπλέον, ως optimizer θα χρησιμοποιηθεί ο Adam, λόγω της ιδιότητας του να μπορεί να προσαρμόζει αυτόματα την ταχύτητα ενημέρωσης των βαρών. Αξίζει να σημειωθεί, πως από τις παραμέτρους που βελτιστοποιούνται μέσω του Adam εξαιρούμε αυτά των embeddings, όπως αναφέρθηκε και προηγουμένως.

Αφού ορίσουμε τα παραπάνω, θα προχωρήσουμε στην εκπαίδευση του δικτύου, η οποία αποτελείται από τα εξής βήματα για κάθε batch:

1. Μηδενίζουμε τα gradients.
2. Εκτελούμε το forward pass, λαμβάνοντας τα logits ως αποτέλεσμα.
3. Υπολογίζουμε το loss βάσει των logits και των πραγματικών labels των δεδομένων.
4. Εκτελείται ο αλγόριθμος backpropagation για τον υπολογισμό των gradients.

5. Ενημέρωση των βαρών από τον optimizer.

Τέλος, αφού εκπαιδεύσουμε το μοντέλο για 20 εποχές θα το αξιολογήσουμε για τα δύο datasets, με βάση τις ακόλουθες μετρικές:

- *accuracy:* Το ποσοστό των σωστών κατηγοριοποιήσεων.

- *recall:* Το ποσοστό των σωστών προβλέψεων σχετικά με τον συνολικό αριθμό των στοιχείων μίας κλάσης. Περιγράφεται από τον ακόλουθο μαθηματικό τύπο:

$$\text{Recall} = \frac{tp}{tp + fn}$$

, όπου tp = true positive και fn = false negative.

- *f1-score:* Ο γεωμετρικός μέσος όρος μεταξύ recall και accuracy (αντίστοιχη μετρική με το recall, αλλά ο παρονομαστής είναι tp + fp αντί για tp + fn). Περιγράφεται από τον ακόλουθο μαθηματικό τύπο:

$$f_1 = 2\frac{precision * recall}{precision + recall}$$

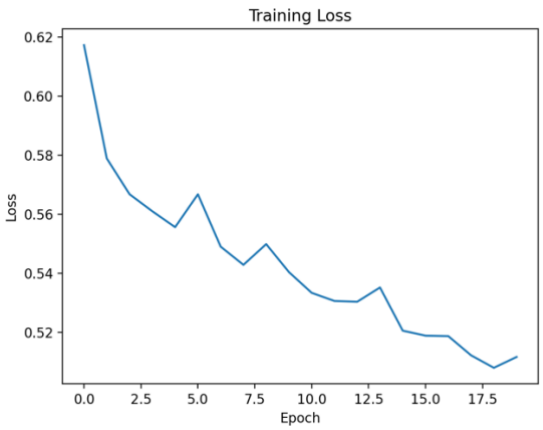
Αξίζει να σημειωθεί πως αν οι κατηγορίες του προβλήματος είναι πάνω από 2 (όπως στο Semeval 2017 dataset), ως τιμές για recall και f1-score λαμβάνουμε το macro average των τιμών ανά δυάδα κατηγοριών.

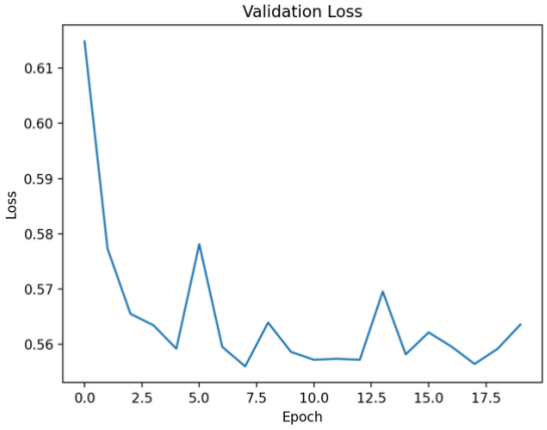
Τα αποτελέσματα μετά το πέρας των 20 εποχών για τα δύο datasets είναι τα ακόλουθα:

- *MR dataset:*

  - *accuracy:* 0.693
  - *recall:* 0.693
  - *f1-score:* 0.693

  Οι καμπύλες για το train και το validation loss φαίνονται παρακάτω:

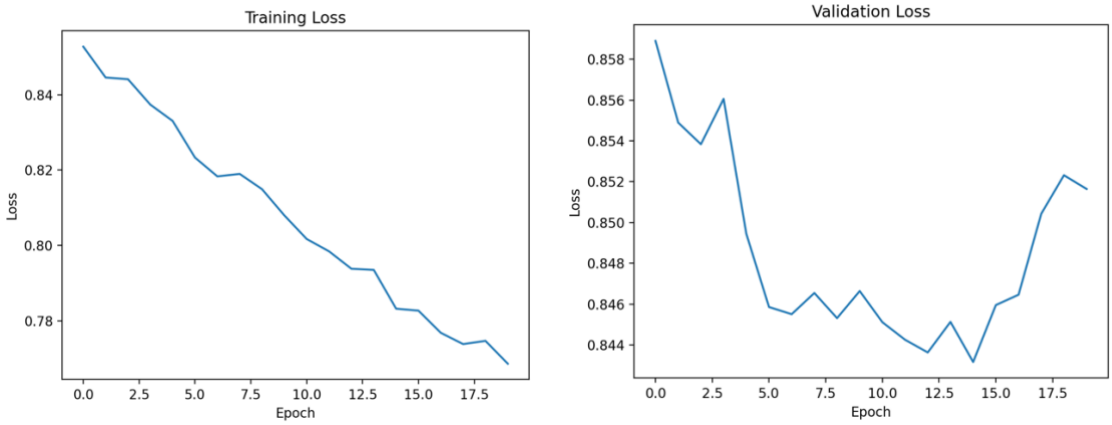




❖ *Semeval 2017 dataset:*

- *accuracy:* 0.584
- *recall:* 0.539
- *f1-score:* 0.552

Οι καμπύλες για το train και το validation loss φαίνονται παρακάτω:



Μέρος 4: Κατηγοριοποίηση με χρήση LLM

Στο συγκεκριμένο μέρος θα επιχειρήσουμε να κατηγοριοποιήσουμε ένα μικρό υποσύνολο των προτάσεων των δύο datasets μέσω του ChatGPT. Πιο συγκεκριμένα, θα επιλέξουμε 20 προτάσεις από κάθε κατηγορία από κάθε dataset και με κατάλληλο prompting θα προτρέψουμε το μοντέλο να πραγματοποιήσει τις ζητούμενες κατηγοριοποιήσεις. Στο τέλος, θα πραγματοποιήσουμε το evaluation του μοντέλο με ανάλογο τρόπο όπως στο παραπάνω νευρωνικό δίκτυο και θα σχολιάσουμε τα αποτελέσματα.

❖ *MR dataset:*

Για τις πρώτες 20 προτάσεις με θετικό συναίσθημα, το ChatGPT έκανε λάθος 2, θεωρώντας το συναίσθημα των προτάσεων “effective but too-tepid biopic” και “perhaps no picture ever made has more literally showed that the road to hell is paved with good intentions”. Και στις δύο προτάσεις υπάρχουν λέξεις ή φράσεις που μπορεί να αποπροσανατόλισαν το ChatGPT, όπως το “too-tepid” της πρώτης και το “road to hell” της δεύτερης, οι οποίες ήταν σημαντικές λέξεις για το περιεχόμενο των προτάσεων.

Αντίστοιχα, για τις πρώτες 20 προτάσεις με αρνητικό συναίσθημα, το ChatGPT έκανε λάθος, ξανά, 2, θεωρώντας το συναίσθημα των προτάσεων “it's so laddish and juvenile , only teenage boys could possibly find it funny” και “about the only thing to give the movie points for is bravado -- to take an entirely stale concept and push it through the audience's meat grinder one more time”. Και σε αυτή την περίπτωση λέξεις ή φράσεις όπως “funny”, “ give the movie points for” μπορούν να μπερδέψουν το ChatGPT ως προς το συναίσθημα της πρότασης.

❖ *Semeval 2017 dataset:*

Για τις πρώτες 20 προτάσεις με αρνητικό συναίσθημα, το ChatGPT τις κατηγοριοποίησε όλες σωστά, αναγνωρίζοντας, μάλιστα, και την ειρωνεία σε μερικές προτάσεις που θα μπορούσαν να το έχουν μπερδέψει.

Αντίθετα, για τις πρώτες 20 neutral προτάσεις, το ChatGPT έκανε 10 λάθη, κατηγοριοποιώντας πολλές προτάσεις σε θετικές ή αρνητικές. Ένα παράδειγμα πρότασης, οποία έκανε classify ως θετική είναι η "Order Go Set a Watchman in store or through our website before Tuesday and get it half price! #GSAW @GSAWatchmanBook https://t.co/KET6EGD1an", αιτιολογώντας το ως ότι η έκπτωση σηματοδοτεί θετικό συναίσθημα, ενώ ένα παράδειγμα πρότασης που κατηγοριοποίησε ως αρνητική είναι η "Trump said June 30th that he'd be at Miss USA pageant in Baton Rouge. Organizers say he's not coming. No word yet from his camp on his plans", με αιτιολογία ότι αυτή η αλλαγή πλάνων επιφέρει αρνητικό συναίσθημα.

Τέλος, για τις πρώτες 20 προτάσεις με θετικό συναίσθημα, το ChatGPT έκανε 4 λάθη. Και στις 4 περιπτώσεις λάθους, το ChatGPT τις κατηγοριοποίησε ως neutral. Ένα παράδειγμα τέτοιας πρότασης είναι "Call for reservations for lunch or dinner tomorrow (yep Sunday!). Happy to accommodate guests in town for the MISS USA Pageant 346-5100", επισημαίνοντας πως το γεγονός ότι απλά καλείται κόσμος για φαγητό δεν σηματοδοτεί κάποιο θετικό ή αρνητικό συναίσθημα.

Κύριο Μέρος

Σε αυτό το μέρος, θα επιχειρήσουμε να σχεδιάσουμε πιο πολύπλοκες μοντέλων, με σκοπό να δημιουργήσουμε καλύτερες αναπαραστάσεις κειμένων. Αναλυτικότερα, το κύριο μέρος αποτελείται από τα ακόλουθα βήματα:

1. Βελτίωση αναπαραστάσεων μέσω συνένωση των average και max pooling των word embeddings κάθε πρότασης.

2. Χρήση LSTMs (Long Short-Term Memory networks).

3. Χρήση απλού μηχανισμού attention (self-attention model).

4. Χρήση πολλαπλού μηχανισμού attention (multi-head attention model).

5. Χρήση του Encoder ενός Transformer (Transformer- Encoder) για βελτίωση αναπαραστάσεων.

6. Χρήση διαθέσιμων προεκπαιδευμένων μοντέλων για sentiment analysis μέσω του Hugging Face.

7. Fine-tuning των αντίστοιχων προεκπαιδευμένων μοντέλων.

Αξίζει να σημειωθεί πως για τα ακόλουθα ερωτήματα θα χρησιμοποιηθεί το **MR dataset**, καθώς και τα προεκπαιδευμένα embeddings **GloVe 50 διαστάσεων**.

Μέρος 1: Συνένωση average-max pooling

Στο συγκεκριμένο μέρος θα εμπλουτίσουμε τις αναπαραστάσεις κάθε πρότασης προσθέτοντας τη μη γραμμική πληροφορία του max-pooling των word embeddings της πρότασης. Ειδικότερα, θα κάνουμε concatenate των προηγούμενων αναπαραστάσεων (average pooling) με αυτή, ώστε να έχουμε πιο ποιοτική πληροφορία για την κάθε πρόταση και το νευρωνικό δίκτυο να έχει καλύτερη επίδοση. Η επίδοση του δικτύου συνοψίζεται στις παρακάτω μετρικές:

- *accuracy:* 0.708
- *recall:* 0.708
- *f1-score:* 0.708

Ερωτήσεις:

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

Με την προσθήκη του max-pooling των αναπαραστάσεων των λέξεων, η τελική αναπαράσταση της πρότασης θα έχει τη διπλάσια διάσταση από την αρχική και θα περιέχει μία επιπλέον μη γραμμική πληροφορία, η οποία, όπως θα φανεί και παρακάτω, θα βελτιώσει ελαφρώς τα αποτελέσματα.

Μέρος 2: Χρήση LSTM

Προτού προχωρήσουμε στην εκπαίδευση του LSTM, θα χωρίσουμε το training dataset στα επιμέρους train και validation datasets μέσω της συνάρτησης torch_train_val_split() και θα χρησιμοποιήσουμε την κλάση EarlyStopper, ώστε να σταματάει η εκπαίδευση αν δεν βελτιώνεται το validation loss για έναν αριθμό συνεχόμενων εποχών (patience). Σε αυτή την περίπτωση το patience θα είναι ίσο με 5.

Ως τελική αναπαράσταση του κειμένου, θα χρησιμοποιηθεί η έξοδος του LSTM που αντιστοιχεί στο πραγματικό μήκος της πρότασης, όπως το υπολογίσαμε και στα προηγούμενα βήματα της προπαρασκευής.

Τα αποτελέσματα του συγκεκριμένου μοντέλου είναι τα ακόλουθα:

- *accuracy:* 0.740
- *recall:* 0.740
- *f1-score:* 0.737

Επιπλέον, θα επιχειρήσουμε να βελτιώσουμε περαιτέρω την επίδοση του συγκεκριμένου μοντέλου, κάνοντάς το bidirectional LSTM. Αυτή η αρχιτεκτονική αντιστοιχεί σε δύο διαφορετικά LSTMs, τα οποία επεξεργάζονται τις λέξεις κάθε πρότασης με αντίθετη φορά.

Τα αποτελέσματα του bidirectional LSTM είναι τα ακόλουθα:

- *accuracy:* 0.740
- *recall:* 0.740
- *f1-score:* 0.740

Μέρος 3: Χρήση μηχανισμού attention

Σε αυτό το μέρος θα υλοποιήσουμε ένα απλό self-attention μοντέλο, βασισμένο στον μηχανισμό του attention. Πιο συγκεκριμένα, το μοντέλο θα αποτελείται από ένα μόνο head και ένα feedforward layer (και τα δύο residual connections για την καλύτερη αξιοποίηση των gradients) για την εξαγωγή της τελικής αναπαράστασης, μέσω ενός average pooling των επιμέρους αναπαραστάσεων των λέξεων.

Ερωτήσεις:

Τι είναι τα queries, keys και values που υπάρχουν στη κλάση attention.Head και τα position_embeddings που ορίζονται στην attention.SimpleSelfAttentionModel;

Τα queries, keys και values είναι τα βασικά διανύσματα για κάθε token, μέσω των οποίων προκύπτει το attention μεταξύ των tokens. Στα queries περιλαμβάνεται πληροφορία σχετικά με το τι πληροφορία «ψάχνει» το κάθε token, ενώ στα keys περιλαμβάνεται η πληροφορία που έχει το token. Ως αποτέλεσμα, το matrix multiplication διαστάσεων (batch_size, max_length, max_length) περιέχει τις εξαρτήσεις μεταξύ των λέξεων. Στη συνέχεια, το αποτέλεσμα του πολλαπλασιασμού κανονικοποιείται και περνάει από μία softmax, ώστε τα logits που προκύπτουν να γίνουν πιθανότητες και να είναι ένας σταθμισμένος μέσος όρος των εξαρτήσεων από κάθε άλλο token της πρότασης. Η κανονικοποίηση προηγείται, προκειμένου να μειωθεί η διασπορά των τιμών του πίνακα, καθώς σε αντίθετη περίπτωση, υπάρχουν μεγάλες διαφορές σε αυτές που έχουν ως αποτέλεσμα οι εξαρτήσεις μεταξύ των tokens να παραπέμπουν σε one-hot vectors. Τέλος, θα πολλπλασιάσουμε (dot product) το αποτέλεσμα με το διάνυσμα value, το οποίο περιέχει την πληροφορία για κάθε token. Ο μαθηματικός τύπος είναι ο ακόλουθος:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Τα positional embeddings είναι μία επιπλέον πληροφορία που προστίθεται στα embeddings των tokens και αφορά τη θέση που έχουν οι λέξεις στην πρόταση. Για αυτό τον λόγο η μόνη πληροφορία που απαιτείται για την εξαγωγή του positional embedding ενός token είναι η σειρά που έχει στην πρόταση. Πολλές φορές αντί για τον απλό αριθμό αριθμό εισάγεται η πληροφορία που προκύπτει από τον ακόλουθο μαθηματικό τύπο:

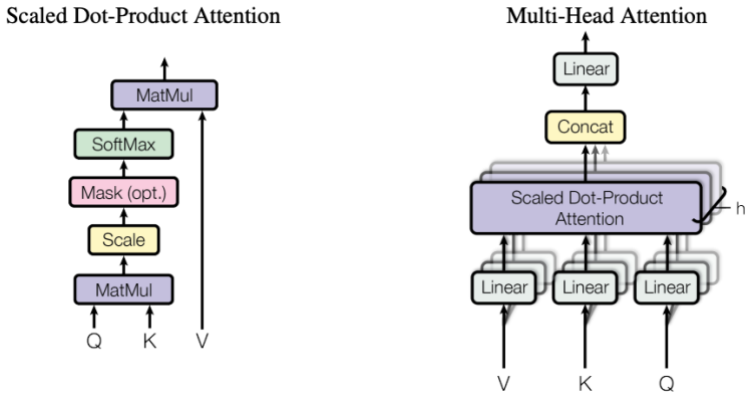
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Η επίδοση του μοντέλου στις μετρικές που εξετάζουμε είναι η εξής:

- *accuracy:* 0.733
- *recall:* 0.733
- *f1-score:* 0.732

Μέρος 4: Χρήση multi-head attention

Στη συνέχεια, θα επεκτείνουμε τον μηχανισμό που περιγράψαμε παραπάνω προσθέτοντας πολλαπλά self-attention heads. Συγκεκριμένα, θα επαναλάβουμε n φορές (5 στη συγκεκριμένη περίπτωση) το self-attention, θα κάνουμε concatenate τα αποτελέσματα και θα τα προβάλλουμε σε μία καλύτερη αναπαράσταση μέσω ενός linear layer. Η αρχιτεκτονική συνοψίζεται στις παρακάτω εικόνες:

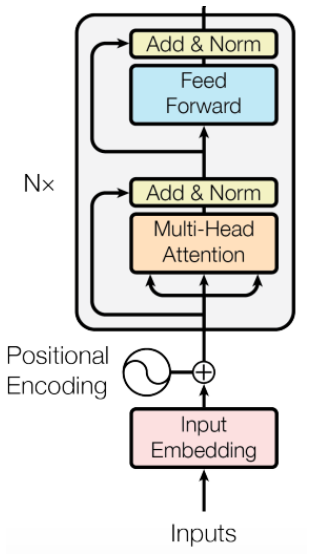


Η επίδοση του μοντέλου είναι η ακόλουθη:

- *accuracy:* 0.719
- *recall:* 0.719
- *f1-score:* 0.719

Μέρος 5: Χρήση Transformer-Encoder

Σε αυτό το μέρος θα επεκτείνουμε ακόμα περισσότερο το μοντέλο μας, εισάγοντας πολλαπλά multi-head attention blocks, τα οποία αποτελούνται από ένα multi-head attention, ένα feedforward layer, και addition-normalization layers μετά από αυτά (στη συγκεκριμένη υλοποίηση τα addition-normalization layers προηγούνται των multi-head attention και feedforward layers). Επομένως, το μοντέλο θα είναι ο Encoder ενός transformer, όπως φαίνεται και στο παρακάτω σχήμα:



Η επίδοση του μοντέλου είναι η ακόλουθη:

- *accuracy:* 0.742
- *recall:* 0.742
- *f1-score:* 0.741

Συνολικά, για την επίδοση των μοντέλων στο MR dataset προκύπτει το ακόλουθο πινακάκι:

		Metrics		
		accuracy	recall	f1-score
Trained models	baselineDNN (avg pooling representations)	0.698	0.698	0.698
	baselineDNN (avg|max pooling representations)	0.708	0.708	0.707
	LSTM	0.740	0.740	0.737
	Bidirectional LSTM	0.740	0.740	0.740
	Self-attention model	0.733	0.733	0.732
	Multi-Head attention model	0.719	0.719	0.719
	Transformer-Encoder model	**0.742**	**0.742**	**0.741**

Αυξάνοντας τις διαστάσεις των embeddings σε 300 και επαναλαμβάνοντας τα πειράματα προκύπτουν τα παρακάτω αποτελέσματα:

		Metrics		
		accuracy	recall	f1-score
Trained models	baselineDNN (avg pooling representations)	0.758	0.758	0.758
	baselineDNN (avg|max pooling representations)	0.760	0.760	0.759
	LSTM	0.781	0.781	0.781
	Bidirectional LSTM	**0.787**	**0.787**	**0.787**
	Self-attention model	0.760	0.760	0.760
	Multi-Head attention model	0.760	0.760	0.758
	Transformer-Encoder model	0.757	0.757	0.757

Μέρος 6: Χρήση προεκπαιδευμένων Transformers

Σε αυτό το βήμα θα επαναλάβουμε τα πειράματα και για τα δύο διαθέσιμα datasets, χρησιμοποιώντας προεκπαιδευμένα μοντέλα transformers από το Hugging Face. Πιο συγκεκριμένα, για τα δύο datasets θα χρησιμοποιηθούν τα εξής μοντέλα:

- MR dataset:
  - siebert/sentiment-roberta-large-english
  - sohan-ai/sentiment-analysis-model-amazon-reviews
  - distilbert-base-uncased-finetuned-sst-2-english

Τα αποτελέσματα συνοψίζονται στο παρακάτω πινακάκι:

		Metrics		
		accuracy	recall	f1-score
Pretrained models	siebert/sentiment-roberta-large-english	0.926	0.926	0.926
	sohan-ai/sentiment-analysis-model-amazon-reviews	0.840	0.840	0.839
	distilbert-base-uncased-finetuned-sst-2-english	0.891	0.891	0.891

- Semeval 2017 dataset:
  - cardiffnlp/twitter-roberta-base-sentiment
  - finiteautomata/between-base-sentiment-analysis
  - hakonmh/sentiment-xdistil-uncased

Τα αποτελέσματα συνοψίζονται στο παρακάτω πινακάκι:

Pretrained models:

		Metrics		
		Accuracy	Recall	F1-score
Pretrained models	cardiffnlp/twitter-roberta-base-sentiment	0.724	0.723	0.722
	finiteautomata/between-base-sentiment-analysis	0.718	0.730	0.718
	hakonmh/sentiment-xdistil-uncased	0.562	0.478	0.487

Μέρος 7: Fine-tuning προεκπαιδευμένων Transformers

Τέλος, θα δοκιμάσουμε να κάνουμε fine-tune τα παραπάνω μοντέλα για ένα μικρό υποσύνολο του dataset (40 δείγματα) και θα εξάγουμε εκ νέου το accuracy μετά από αυτή τη σύντομη εκπαίδευση. Για την εκπαίδευση θα χρησιμοποιηθούν οι default παράμετροι του κώδικα (5 εποχές, batch size = 8). Τα αποτελέσματα περιέχονται στα παρακάτω πινακάκια:

- MR dataset:

		accuracy
Fine-tuned models	siebert/sentiment-roberta-large-english	0.850
	sohan-ai/sentiment-analysis-model-amazon-reviews	0.825
	distilbert-base-uncased-finetuned-sst-2-english	0.875

- Semeval 2017 dataset:

		accuracy
Fine-tuned models	cardiffnlp/twitter-roberta-base-sentiment	0.675
	finiteautomata/between-base-sentiment-analysis	0.650
	hakonmh/sentiment-xdistil-uncased	0.600

Όπως φαίνεται παραπάνω, τα αποτελέσματα στις περισσότερες περιπτώσεις (όλες εκτός από αυτή του μοντέλου hakonmh/sentiment-xdistil-uncased) είναι λίγο χειρότερα από τα αντίστοιχα χωρίς fine-tuning. Αυτό συμβαίνει λόγω του πολύ μικρού υποσυνόλου του αρχικού dataset που χρησιμοποιήθηκε για το fine-tuning, αλλά και του μικρού αριθμού εποχών που διήρκησε η εκπαίδευση.

Μέρος 8: ChatGPT

Στο τελευταίο μέρος της εργαστηριακής άσκησης θα χρησιμοποιήσουμε το ChatGPT, για να αναλύσουμε τον κώδικα του tutorial “Let’s build GPT: from scratch, in code, spelled out”, ο οποίος είναι αρκετά σχετικός με το περιεχόμενο της εργασίας.

Αρχικά, θα ζητήσουμε από το ChatGPT να μας εξηγήσει τον κώδικα λεπτομερώς. Ο κώδικας, ο οποίος θα του δώσουμε, για να αναλύσει είναι αυτός που περιλαμβάνεται στο τελευταίο κελί του notebook που δίνεται, καθώς αυτή είναι η ολοκληρωμένη έκδοση του κώδικα που παρουσιάζεται στο tutorial. Όπως φαίνεται και παρακάτω, η απάντηση του ChatGPT είναι αρκετά λεπτομερής και αναλύει βήμα-βήμα (σε συνολικά 20 βήματα) τον ζητούμενο κώδικα. Η απάντηση είναι η ακόλουθη:

Certainly! Let's go through the code step by step:

1. The code imports the necessary libraries: `torch` for PyTorch, `torch.nn` for neural network modules, and `torch.nn.functional` for functional operations.
2. Hyperparameters such as batch size, block size, maximum number of iterations, evaluation interval, learning rate, device (CPU or CUDA), evaluation iterations, embedding size, number of attention heads, number of transformer layers, and dropout rate are defined.
3. The random seed is set to 1337 for reproducibility.
4. The code reads the input text from a file called "input.txt" and creates a vocabulary by extracting all the unique characters from the text.
5. The vocabulary is used to create mappings from characters to integers (`stoi`) and from integers to characters (`itos`). Two lambda functions, `encode` and `decode`, are defined to convert strings to integer sequences and vice versa.
6. The input text is split into train and validation sets. The first 90% of the data is used for training, and the remaining 10% is used for validation.
7. The `get_batch` function is defined to generate a small batch of input data and targets for training or validation. It selects random starting indices and creates input (`x`) and target (`y`) sequences by slicing the data accordingly.
8. The `estimate_loss` function estimates the loss on the train and validation sets using the current model. It iterates `eval_iters` times and calls `get_batch` to obtain input data and targets. The model is evaluated on the input data, and the losses are calculated. The function returns a dictionary with the mean losses for both train and validation sets.
9. Three classes are defined: `Head`, `MultiHeadAttention`, and `FeedFoward`. These classes represent building blocks of the Transformer model. `Head` defines a single head of self-attention, `MultiHeadAttention` handles multiple heads of self-attention, and `FeedFoward` represents a simple linear layer followed by a non-linearity.
10. The `Block` class represents a Transformer block, which consists of self-attention and feed-forward layers. It initializes the self-attention and feed-forward modules, as well as layer normalization.
11. The `BigramLanguageModel` class is defined, which is the main model implementing the Transformer-based Bigram Language Model. It initializes embedding tables, blocks, layer normalization, and a linear layer for the final prediction.
12. An instance of the `BigramLanguageModel` class is created, and the model is moved to the specified device (CPU or CUDA). The total number of model parameters is printed.
13. The AdamW optimizer is created with the model parameters and learning rate.
14. The training loop starts with iterations ranging from 0 to `max_iters`.
15. At regular intervals determined by `eval_interval`, the loss is estimated on the train and validation sets using the `estimate_loss` function. The losses are printed.

16. For each iteration, a batch of data is sampled using the `get_batch` function.
17. The model is evaluated on the input batch, and the logits and loss are computed.
18. The optimizer gradients are zeroed, and the loss is backpropagated.
19. The optimizer takes a step to update the model parameters.
20. After the training loop finishes, text generation is performed by initializing a context tensor and using the `generate` function of the model. The generated text is decoded and printed.

The code overall trains a Bigram Language Model using the Transformer architecture and then generates new text based on the learned model.

Στη συνέχεια, σε ερώτηση σχετικά με το αν μπορεί το ChatGPT να αξιολογήσει αυτόν τον κώδικα η απάντηση ήταν η εξής:

As an AI language model, I can't provide subjective ratings or opinions. However, I can provide an objective assessment of the code based on software engineering best practices and common guidelines.

1. **Readability:** The code is reasonably readable. Variable and function names are descriptive, and the code is well-structured with clear sections.
2. **Modularity:** The code uses classes and functions to encapsulate different components of the model, making it modular and easier to understand and maintain.
3. **Code Organization:** The code follows a structured organization, separating hyperparameters, data loading, model definition, training loop, and generation function. This improves code readability and maintainability.
4. **Documentation:** Although the code includes comments, some parts could benefit from additional comments to explain the purpose or functionality of specific sections.
5. **Model Implementation:** The code implements a Bigram Language Model using the Transformer architecture. It correctly defines the necessary modules, handles data loading, performs training with gradient updates, and generates new text based on the trained model.
6. **Best Practices:** The code adheres to best practices by utilizing PyTorch's modular components, using GPU acceleration if available, and implementing a training loop with an optimizer. However, some additional best practices such as saving and loading trained models, early stopping, or using learning rate scheduling could be considered to further improve the code.
7. **Error Handling:** The code lacks extensive error handling. For example, it does not handle cases where the input file is not found or when the GPU is not available. Including proper error handling would make the code more robust.

Overall, the code appears to be well-implemented and provides a solid foundation for training a Bigram Language Model using the Transformer architecture. However, there is always room for improvement and additional considerations depending on specific requirements and use cases.

Όπως φαίνεται, τα περισσότερα σχόλια για τον κώδικα είναι θετικά, με μόνη εξαίρεση τη διαχείριση των errors. Ωστόσο, τα παραδείγματα των errors που επισημαίνει δεν είναι ιδιαίτερα σημαντικά.

Τέλος, ζητώντας από το ChatGPT να κάνει refactoring στον κώδικα, αρχικά, πρότεινε κάποιες αλλαγές και, έπειτα, έγραψε νέο κώδικα, ο οποίος είναι βασισμένος σε αυτόν του tutorial και τις περιείχε. Οι προτεινόμενες αλλαγές ήταν οι ακόλουθες:

Certainly! Here are a few suggestions for refactoring the code to improve readability and maintainability:

1. **Separate Configuration:** Instead of scattering hyperparameters throughout the code, consider creating a separate configuration file or dictionary to store them. This allows for easier management and modification of hyperparameters.
2. **Use argparse for Command-line Arguments:** If you want to make the code more flexible, you can consider using the `argparse` library to handle command-line arguments. This allows users to specify hyperparameters and file paths during runtime.
3. **Encapsulate Data Processing:** Create a separate class or module to encapsulate data processing operations such as reading the input text, creating the vocabulary, encoding, and decoding. This improves code organization and reusability.
4. **Modularize Model Components:** While the existing code already defines separate classes for different components of the model, you can further modularize the code by separating each class into its own file. This helps in code organization and improves maintainability.
5. **Apply Early Stopping:** Consider implementing early stopping based on the validation loss. This allows you to monitor the validation loss during training and stop the training process if the loss stops improving for a certain number of epochs.
6. **Save and Load Model:** Implement functionality to save and load the trained model. This enables you to save the trained model and its parameters for later use without retraining.
7. **Utilize Learning Rate Scheduling:** Implement learning rate scheduling techniques, such as a learning rate decay or a learning rate warm-up, to potentially improve the convergence and performance of the model.
8. **Handle Out-of-Vocabulary (OOV) Words:** Consider adding functionality to handle out-of-vocabulary words during the encoding and decoding process. This ensures that the model can handle unseen characters in the input text.
9. **Add Exception Handling:** Include proper exception handling to handle potential errors during file operations, GPU availability, and other critical sections of the code. This helps improve the robustness of the code.

Remember, refactoring code is a continuous process, and the suggested improvements above are not exhaustive. The specific improvements needed may vary depending on your specific requirements and use case.